UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

HOCHSCHULE HEILBRONN
TECHNIK WIRTSCHAFT INFORMATIK

dkfz.

DEUTSCHES
KREBSFORSCHUNGSZENTRUM
IN DER HELMHOLTZ-GEMEINSCHAFT

Integration von LIBSVM in MITK und Test anhand einer automatischen Lebertumorsegmentierung

Bachelor-Thesis

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.) der Medizinischen Fakultät
der Universität Heidelberg im Studiengang
Medizinische Informatik

vorgelegt von

Alexander Tschlatscher, 26.Februar 2016

Referent: Dr. rer. nat. Klaus Hermann Maier-Hein, DKFZ

Koreferent: Prof. Dr. sc. hum. Rolf Bendl, Hochschule Heilbronn

Betreuer: M.Sc. Michael Götz, DKFZ

EIDESSTATTLICHE ERKLÄRUNG

Hiermit versichere ich, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln (einschließlich elektronischer Hilfsmittel) angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht. Die vorliegende Arbeit wurde keiner anderen Prüfungsbehörde vorgelegt.

....

(Ort, Datum) (Unterschrift)

DANKSAGUNG

Ich möchte mich herzlich bedanken bei meinem betreuenden Referenten Dr. Klaus Hermann Maier-Hein, er hatte jederzeit und für jedes Problem ein offenes Ohr sowie super Ideen zur Weiterentwicklung der Thesis. Ein ganz besonderer Dank geht an meinen Betreuer Herrn Michael Götz, der mich während der Bearbeitung meiner Thesis sehr kompetent und umfassend unterstützt hat. Durch sein sehr interdisziplinäres und breites Fachwissen konnte ich in den Bereichen Softwareentwicklung, Mathematik, Data Mining und medizinische Bildverarbeitung noch sehr viel dazulernen. Ein weiterer Dank gebührt Herrn Jonas Cordes. Er hat mich während der Vorbereitungsphase für diese Bachelorthesis im Bereich der Softwareentwicklung mit C++ sehr freundlich und kompetent unterstützt. Auch bei meinem Koreferenten Herrn Prof. Dr. Rolf Bendl möchte ich mich sehr bedanken. Durch seine sehr interessanten Vorlesungen und Laborübungen im Bereich der medizinischen Bildverarbeitung fing ich an, mich sehr für diesen Themenbereich zu interessieren. Zudem hatte sich Herr Bendl während meines Studiums immer Zeit genommen, auftauchende Fragen ausführlich und kompetent zu beantworten. Zuletzt möchte ich mich bei meiner Familie bedanken, die mich während meines Studiums sehr tatkräftig unterstützt hat und mir es somit ermöglichte, soweit zu kommen.

KURZFASSUNG

In der heutigen Zeit ist es auch für Ärzte nicht immer trivial, diagnostisch präzise Entscheidungen auf Grund von modernsten medizinischen bildgebenden Verfahren zu treffen. Die vorliegende Bachelor-Thesis befasst sich damit wie man eine Softwareanwendung mit Hilfe der Support Vector Machine (SVM), eine Technik des maschinellen Lernens, in ein bestehendes Framework wie z.B. MITK implementieren kann, um so vollautomatische Tests von Lebertumorsegmenierungen durchführen zu können. Durch die zusätzliche Integration einer Testklasse wird die entwickelte SVM validiert, um ein möglichst hohes Klassifikationsergebnis zu erreichen. Die in der Thesis entwickelte Softwarekomponente hat gezeigt, dass eine vollautomatische Segmentierung von Lebertumoren bei Patienten in zufriedenstellendem Maße möglich ist.

ABSTRACT

Nowadays it is not easy for doctors to make accurate diagnostic decisions due to modern medical imaging techniques. This bachelor thesis is focused on how to integrate a software application which uses a technique of machine learning, the support vector machine (SVM), in an existing framework like MITK to make full automatically tests of Livercancersegmentations. With the additionally implementation of a testsuitclass it´s possible to validade the SVM-Application with this Testclass to achieve a very good classification result. The SVM-Application shows, that it´s possible to make full automatically segmentations of livercancer by different patients with satisfied results.

INHALTSVERZEICHNIS I

ABBILDUNGSVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

HCC	Hepatozelluläres Karzinom
SVM	Support Vector Machine
SVC	Support Vector Classification
SVR	Support Vector Regression
CT	Computertomografie
MITK	Medical Imaging Interaction Toolkit
ITK	Insight Segmentation and Registration Toolkit
VTK	Visualization Toolkit
2D	zweidimensional
3D	dreidimensional
RBF	Radial Basis Function
DICOM	Digital Imaging and Communications in Medicine
XML	Extensible Markup Language
DoG	Different of Gaussian

SYMBOLVERZEICHNIS

$f(x,y)$	Bildfunktion
$(x_i\ , y_i)$	Menge von abgebildeten Trainingsmustern
$K(x_i\ , y_i)$	Kernelfunktion
μ	Erwartungswert
σ	Standardabweichung
b	Bias
ω	Normalenvektor
y_i	Klassenlabels
x_i	beobachtete Merkmale
z_i	i-ter Stützvektor
a_i	Gewichte vom i-ten Stützvektor z_i
$\varphi(x)$	Transformationsfunktion
$sign(x)$	reelle Vorzeichenfunktion
$\mathbb{R}$	Die Menge der reellen Zahlen
$\otimes$	Der Faltungsoperator
K	Radiusvektor der Standardabweichung
GN_{xx}	$\frac{\partial^2 GN(x,y)}{\partial x^2}$ die zweite partielle Ableitung nach x
GN_{yy}	$\frac{\partial^2 GN(x,y)}{\partial y^2}$ die zweite partielle Ableitung nach y
GN_{xy}	$\frac{\partial^2 GN(x,y)}{\partial x \partial y}$ oder $\frac{\partial^2 GN(x,y)}{\partial y \partial x}$ gemischt partielle Ableitung

1. Einführung

1.1 Hintergrund und Motivation

In Deutschland zählt das hepatozelluläre Karzinom (HCC) zu den eher seltenen Tumorarten, trotzdem gehört der Lebertumor allgemein zu den zehn häufigsten Krebstodesursachen. Grund dafür ist eine schlechte Prognose des Tumors sowohl bei Männern als auch bei Frauen. Pro Jahr treten in Deutschland 8300 neue Fälle von Patienten mit Lebertumoren auf, „mit annähernd gleicher Zahl von Todesfällen“. [GBEB 13] Statistisch betrachtet liegt das mittlere Erkrankungsalter, für Männer bei 70 Jahren und für Frauen bei 74 Jahren. Im Laufe ihres Lebens erkrankt „einer von 86 Männern und eine von 200 Frauen in Deutschland“ an einem HCC malignen Lebertumor. [GBEB 13] Fast zwei Drittel der malignen Lebertumore (ungefähr 65%) entstehen aus den Hepatozyten und bilden hepatozelluläre Karzinome. Die restlichen 25% der malignen Lebertumore sind sogenannte Cholangoikarzinome aus den intrahepatischen Gallengängen. [GBEB 13]

1.2 Ziele und Problemstellung der Thesis

Primäres Ziel der vorliegenden Bachelorarbeit ist es, eine so genannte Support Vector Machine (SVM), ein Verfahren zur Datenklassifikation, zu implementieren und in das Framework MITK zu integrieren. Die implementierte SVM soll in der Lage sein, vollautomatisch in CT-Bilddatensätzen Lebertumore vom gesunden Lebergewebe bildbasiert zu differenzieren und in den richtigen Bildregionen zu klassifizieren. Für die Implementierung der SVM wird eine Wrapperklasse verwendet, welche in der Lage ist, die LIBSVM in die bestehende Softwareapplikation zu integrieren. Sekundäres Ziel ist es, für die implementierte SVM eine Testklasse zu schreiben, die durch Generierung von verschiedenen Testdatensätzen und Testmethoden diese automatisiert testet und validiert. Wichtig bei der Implementierung der Testklasse ist es darauf zu testen, wie gut die integrierte SVM mit unterschiedlichen Datensätzen klassifiziert. Die entwickelte Softwareanwendung soll anhand eines Real-Life-Testes, also einem Testdurchlauf unter reellen Bedingungen, eine vollautomatische Lebertumorsegmentierung bei drei Patienten durchführen. Bisher wurde dieser Real-Life-Test noch nie durchgeführt und deswegen ist es umso spannender zu sehen, wie die integrierte SVM bei diesem Real-Life-Test abschneidet. Ein daraus resultierendes Problem der Klassifikation von Lebertumoren sind die Atembewegungen der Patienten während der Aufnahme der CT-Bilddatensätze.

2. Grundlagen

In dem folgenden Grundlagenkapitel werden die Themen behandelt, die im Kontext der Bachelorthesis relevant sind. Zuerst wird kurz auf die allgemeinen Grundlagen zur Physiologie der Leber eingegangen. (Kapitel 2.1) Des Weiteren interessiert die Art des Tumors (Kapitel 2.2), die mithilfe des Klassifikators SVM (Kapitel 2.3.3) segmentiert und klassifiziert wird. Anschließend steht der Aufbau von medizinischen Bildern (Kapitel 2.3.1) sowie Bildtexturen im Mittelpunkt der Arbeit (Kapitel 2.3.2), die uns dabei helfen sollen, signifikante Strukturen wie zum Beispiel hepatozelluläre Karzinome, aus einem gegebenen medizinischen CT-Bilddatensatz zu segmentieren. Im darauffolgenden Kapitel 2.3.3.1 wird erläutert, wie die SVM prinzipiell funktioniert. Da die SVM einen hohen Bezug zur Mathematik hat, wird im Kapitel 2.3.3.2 ein kleiner Einblick derjenigen Mathematik gegeben, welche man benötigt, um den Klassifikationsprozess einer SVM zu verstehen. Danach folgt eine Beschreibung der Verwendung von SVM-Parametern in Kapitel 2.3.3.3. und was passiert, wenn die SVM-Parameter variiert werden. Abschließend wird die verwendete Bibliothek LIBSVM (Kapitel 2.4.1) sowie das Framework MITK (Kapitel 2.4.2) erläutert.

2.1 Die anatomischen Grundlagen der Leber

Die Leber (lat. „hepar“) ist mit einem Gewicht von rund 1500 g die größte Drüse des menschlichen Körpers. [HK 07] Zu den wichtigsten Aufgaben der Leber zählen die Produktion von Gerinnungsfaktoren, der Abbau von Medikamenten und Giftstoffen durch die Gallenproduktion und die Ausscheidung der Stoffwechselendprodukte. Über die Pfortader der Leber (lat. „vena portae hepatis“) wird das nährstoffreiche Blut aus dem Darm zugeleitet. Die Aufnahme dieser Nährstoffe erfolgt direkt in der Leber. Verarbeitet werden die Nährstoffe in den Hepatozyten. Somit gehört die Leber zum wichtigsten Verdauungstrakt. [HK 07] Die wesentlichen Funktionen der Leber sind:

- Bildung von Galle und Bilirubin (ein Abbauprodukt des Hämoglobins)
- Entgiftungsfunktion für Alkohol und Medikamente
- Vitaminspeicherung
- Proteinsynthese von Albumin und Gerinnungsfaktoren
- Vielfältige Funktionen im Metabolismus des Fett,- Protein- und Kohlenhydrat-stoffwechsels

[HK 07]

Anatomisch lokalisiert ist die Hauptmasse der Leber unter der rechten Zwerchfell-kuppel im Schutz des knöchernen Brustkorbs. Somit liegen die Leber und die Gal-lenblase, welche an der Leber befestigt ist, intraperitoneal. [HK 07] [HS CW 15] Die Lokalisation der Leber ist in (Abbildung 1) gut ersichtlich.

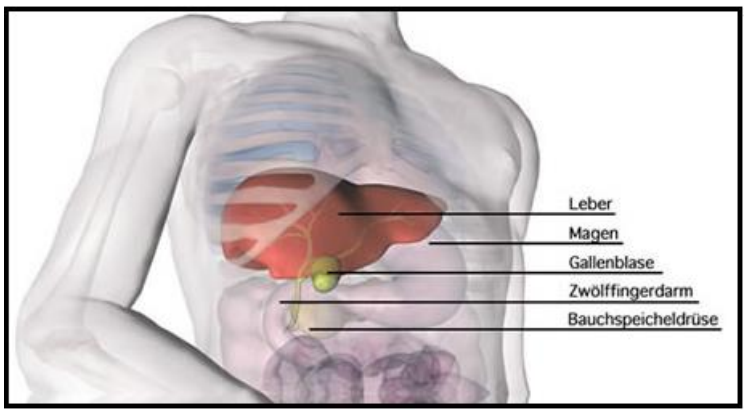


Abbildung 1: Lokalisation der Leber [Apo15]

Bräunlich dargestellt ist die Leber. Die konkave Unterfläche der Leber liegt auf der Niere, dem Magen und dem Zwölffingerdarm.

2.2 Das hepatozelluläre Karzinom

In diesem Kapitel geht es um die Art des Tumors, die das Klassifikationsprogramm in den CT-Bilddatensätzen detektieren und klassifizieren soll, das sogenannte hepatozelluläre Karzinom.(Siehe Abbildung 2) Hepatozelluläre Karzinome sind hochgradig maligne, primäre Lebertumore, welche ihren Ursprung in Hepatozyten der Leber haben und 80% aller primären Lebertumore ausmachen. Im Ranking der weltweit häufigsten malignen Tumore steht das HCC an fünfter Stelle. Die meisten Patienten, die einen HCC haben, erkrankten zuvor an einer Leberzirrhose. Eine Leberzirrhose kann durch progressiven Alkoholkonsum entstehen oder durch die Infektion mit Hepatitis B und C. [LJ 07]

Betrachtet man die auftauchenden Inzidenzen von HCC weltweit, so stellt man fest, dass die Inzidenz in Europa 1 bis 2 Fälle von HCC pro 100 000 Einwohner beträgt und die Inzidenz in Südostasien sowie in Süd-, Nord- und Zentralafrika 100 Fälle pro 100 000 Einwohner zählt. Das Risiko an einem HCC zu erkranken ist für Männer viermal höher als für Frauen. [LJ 07] Das HCC liefert eine unbefriedigende Prognose, da die Mehrheit der am HCC erkrankten Patienten erst im sehr fortgeschrittenen Stadium diagnostiziert wird. [LJ 07]

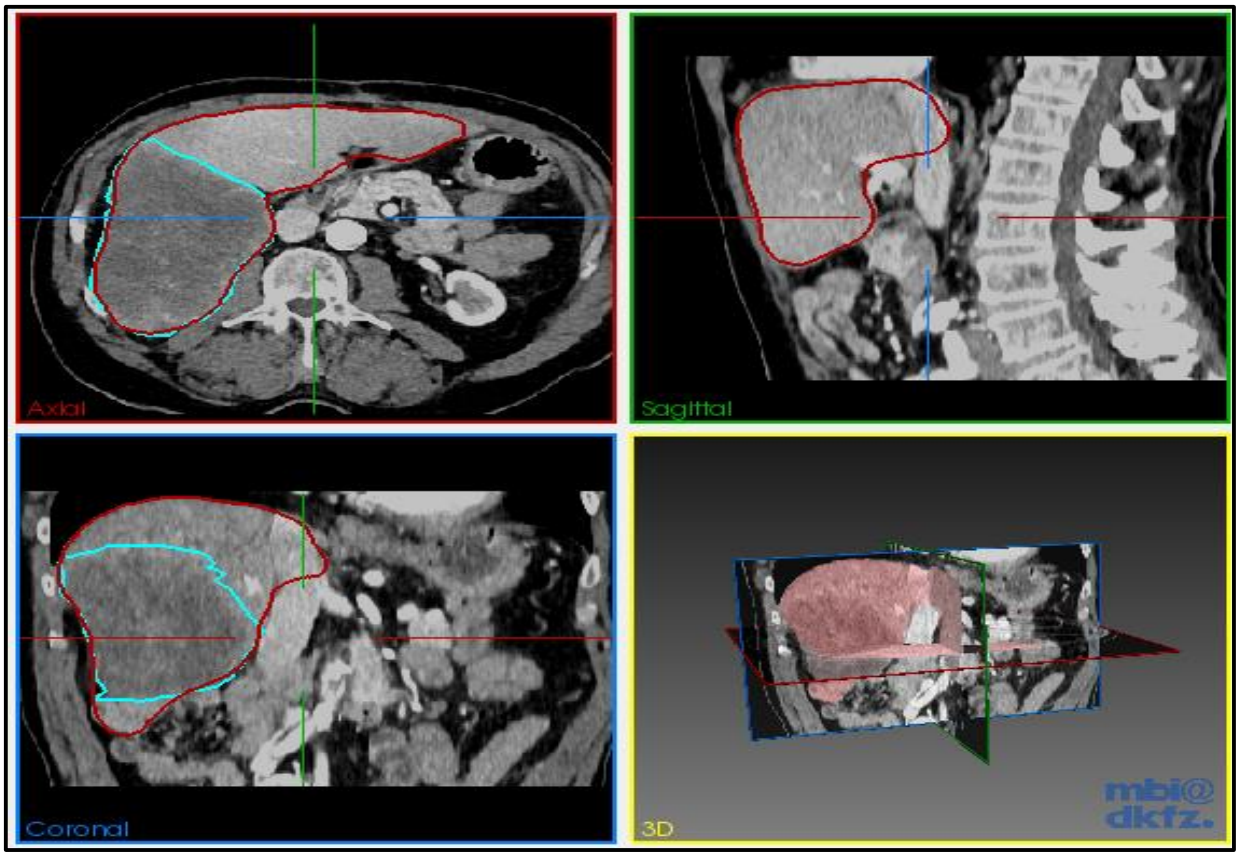


Abbildung 2: Visualisierung von Leber und HCC mit MITK

Rotbräunlich umrandet sieht man die Kontur der Leber. Die Kontur des HCC ist hellblau im Leberbereich dargestellt.

2.3 Klassifikation von medizinischen Bildern

2.3.1 Der Aufbau von medizinischen Bilddaten

In der Bildverarbeitung kann ein 2D Bild als eine zweidimensionale Matrix der Form $f: [0, \dots\dots, M] \times [0, \dots\dots, N] \rightarrow \{0, \dots\dots, g-1\}$ mit $M-1$ Zeilen und $N-1$ Spalten interpretiert werden.
Jedes 2D Bild kann in diese Matrixform der Bildgrauwerte transformiert werden:

$$Bildmatrix: f(x,y) = \begin{pmatrix} f(0,0) & f(1,0) & f(N-1,0) \\ \vdots & \dots & \vdots \\ f(0,M-1) & f(1,M-1) & f(N-1,M-1) \end{pmatrix}, \forall (x,y) \in \mathbb{R}$$

modifiziert nach [HH 09]

Jeder Pixel eines Bildes kann einem Grauwert, Signalwert oder Parameterwert zugeordnet werden. [HH 09] Dabei entspricht $f(x,y)$ der Bildfunktion und g entsprechend den 256 Grauwerten, wenn jedes Voxel als ein Byte gespeichert wird.

In 2D-Bilddatensätzen wird mit Pixelelementen (Abbildung 3a) und in 3D Bilddatensätzen, wie zum Beispiel tomografischen Bildern (Abbildung 3b), mit Voxelelementen gearbeitet. Neben den Bildpunktelementen ist noch ein weiterer Faktor signifikant für die medizinische Bildgebung, die sogenannte Grauwerttiefe. [HH 09] Diese gibt an, „wie fein diskretisiert die Bildfunktionswerte in der Bildmatrix repräsentiert werden." [HH 09] In CT-Bilddatensätzen beträgt die Grauwerttiefe 12 Bit, also $2^{12} = 4096$ Grauwerte und in trivialen schwarz/weiß Bilddaten nur 8 Bit, also $2^8 = 256$ Grauwerte. [HH 09] In dieser Arbeit differenzieren wir zwischen 3 Bildtypen:

Grauwertbild:	CT-, MRT- oder Röntgenbilder mit 4096 Grauwertabstufungen
Parameterbild:	Texturparameterbilder, mit passenden Merkmalen als Parameter
Indexbild:	Labelbilder generiert durch Segmentierungs- oder Klassifizierungsalgorithmen

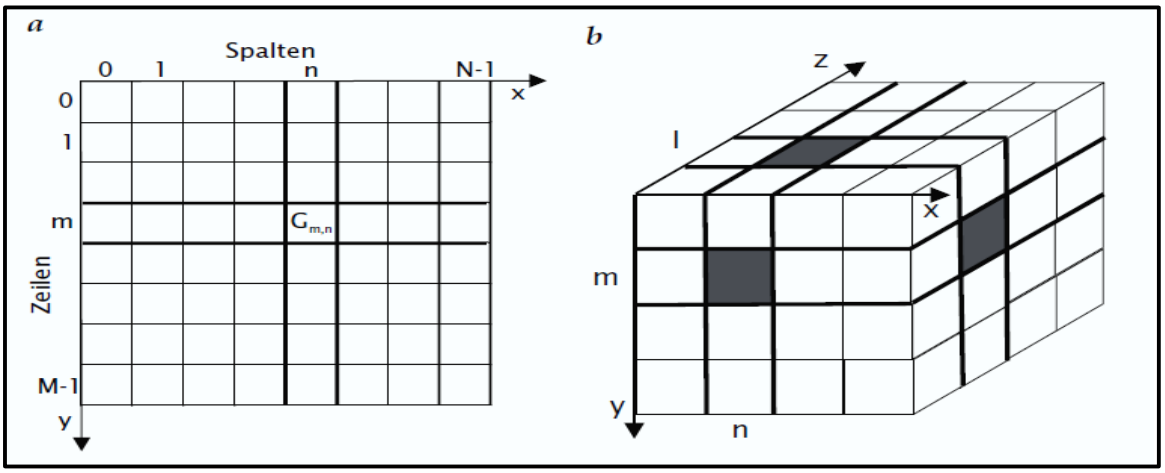


Abbildung 3: Darstellung eines 2D- und 3D-Bildes auf einem Gitter

Jedes (medizinische) digitale Bild kann mit seinen diskreten Werten (Pixel und Voxel) in einer Gitterstruktur repräsentiert werden. [B.Jähne 12]

2.3.2 Texturen

Häufig reichen Grauwerte nicht aus, um morphologische Strukturen wie die der Leber zu klassifizieren. Aus diesem Grunde soll die morphologische Struktur der Leber dem Klassifikator als Textur übergeben (siehe Abbildung 5) werden. In der Bildverarbeitung werden Bildtexturen als Bilder bezeichnet, die auf die Oberfläche eines virtuellen Körpers projiziert werden können. Texturen geben uns Informationen über die räumliche Einteilung von Farbintensitäten oder einer ausgewählten Region in einem Bild. [LGSG 01] Eine Textur besteht aus einer Menge von Texturelementen (engl. „texels“), die entweder in einem sich wiederholenden Muster auftreten oder eine zufällige Anordnung haben. [TKAK 15] Ein Beispiel einer Textur ist in (Abbildung 4) ersichtlich. Gut zu erkennen ist die zufällige, aber strukturierte Anordnung der einzelnen Steine, die als Gesamtbild betrachtet ein Muster ergeben. Hinsichtlich ihres Ursprungs differenzieren wir zwischen zwei Arten von Bildtexturen: Einmal können Texturen künstlich erzeugt werden, wenn wir beispielsweise an ein Schachbrettmuster denken, oder sie können von Natur aus vorhanden sein (siehe Abbildung 4). [TKAK 15]

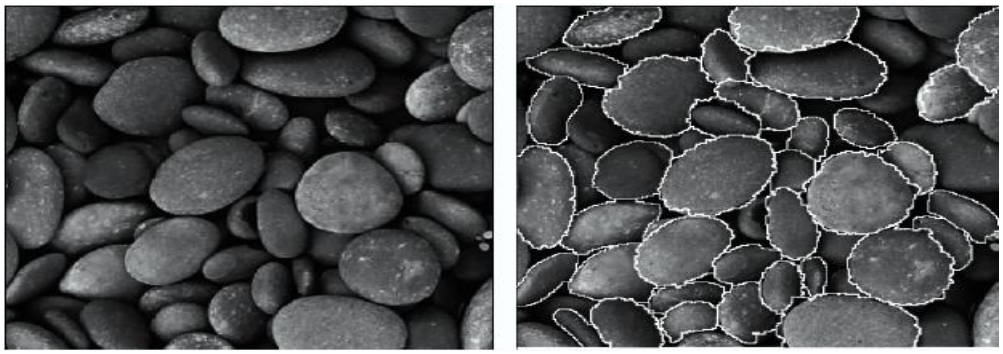


Abbildung 4: Darstellung einer natürlichen Textur

Links in der Abbildung ist eine natürliche Textur visualisiert, auf der rechten Abbildung sieht man weiß umrahmt die einzelnen Pixelelemente der Textur, sogenannte Texels.

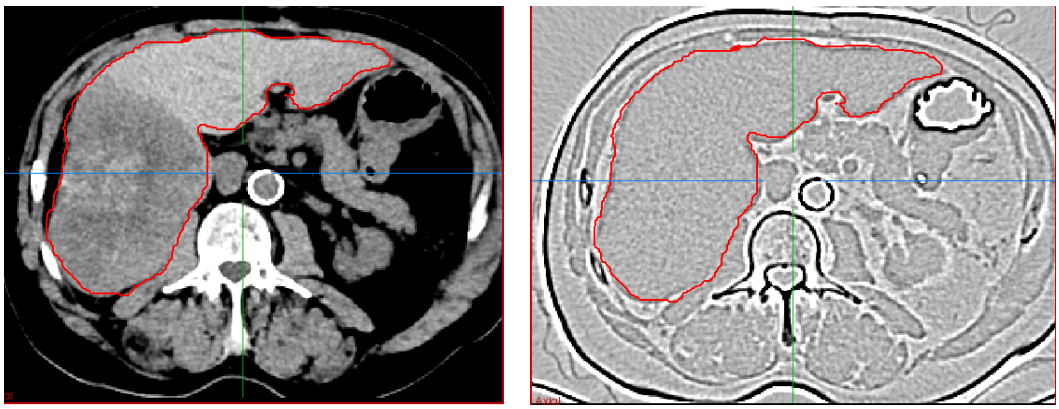


Abbildung 5: Leberbild und eine daraus abgeleitete Texturkarte (mit DoG berechnet)

Die Texturkarte (rechts abgebildet) wurde mithilfe der ersten Ableitung berechnet. Die abgeleiteten Kanten sieht man sehr deutlich im Bild, da die Ableitung der momentanen Änderung der einzelnen Signalwerte im Bild entspricht.

2.3.3 Support Vector Machines (SVM)

2.3.3.1 Funktionsweise einer SVM

SVM ist eine Technik zur Datenklassifikation, kann aber auch zur Regression eingesetzt werden. [ChLin 11]. In dieser Anwendung möchten wir die SVM einsetzen, um in medizinischen Bildern Lebertumore zu klassifizieren. Die Aufgabe der Klassifikation ist es, gegebene Daten Klassen zuzuordnen. Der erste Schritt der Klassifikation mit SVM ist es, ein Modell basierend auf den Trainingsdatensatz, zu berechnen. Im nächsten Schritt wird dann mithilfe des Modells und der Testdatenattribute der noch fehlende Teil des Testdatensatzes prädiziert. Jedem Vektorelement im Vektorraum wird ein spezieller Featuretyp zugeordnet, sodass jeder Vektor einer Beobachtung entspricht. Features können daher z.B. den Grauwerten einer Modalität oder den Mittelwerten der Grauwerte entsprechen. In (Abbildung 6) sind die Features durch die Koordinatenachsen beschrieben. Das bedeutet, die Abszisse ist unser erstes, die Ordinate unser zweites Feature. (Abbildung 6a) zeigt einen Datensatz in so einem Vektorraum, dabei sind zwei Labels vertreten. Dabei repräsentieren die blauen Punkte Gewebe ohne Leberläsionen (z.B. Klasse ohne Tumor) und die orangenen Punkte Gewebe mit Leberläsionen (z.B. Klasse mit Tumor). Festzustellen ist, dass in diesem Fall, eine lineare Separation der Daten durch Hyperebenen, im 2D Raum mit Geraden, möglich ist. [CoVa 95] In beiden Teilabbildungen (Abbildung 6a und 6b) wurden zwei Hyperebenen h1 und h2 eingezeichnet. Damit wir anschließend die optimale Ebene finden wird, ein Abstand (engl. margin) zu den Trainingspunkten der unterschiedlichen Klassen definiert (siehe Abbildung 6b). Die Ebene mit dem maximalen Abstand wird ausgewählt und entspricht der optimalen Hyperebene. Wir stellen fest, dass Hyperebene h1 die Daten besser voneinander trennt als Hyperebene h2 (siehe Abbildung: 6b).

In den meisten Anwendungsfällen ist eine lineare Separation im Eingabevektorraum nicht möglich (siehe Abbildung 7 links). [CoVa 95] Es liegt also ein nicht linear trennbarer Fall vor. Bei der Lösung dieses Falles hilft ein kleiner Trick. Wie in (Abbildung 7) visualisiert kann das Problem gelöst werden, indem die Patientendaten im Eingabevektorraum mit Zuhilfenahme einer Transformationsfunktion $\varphi(x)$ in den Merkmalsvektorraum transformiert werden. [CoVa 95] Hierfür müssen wir den n-dimensionalen Input-Vektor (Abbildung 7 links orangene Kugel im Input-Space, von welcher aus der Pfeil entspringt) in einem N-dimensionalen (höherdimensionalen) Feature-Vektor (Abbildung 7 rechts orangene Kugel im Feature Space auf den der Pfeil zeigt) transformiert werden mit Hilfe einer N-dimensionalen Transformationsfunktion: $\varphi(x)$.

$Transformation: \varphi: \mathbb{R}^n \rightarrow \mathbb{R}^N$:

Wir transformieren φ vom einem reellen n-dimensionalen Vektorraum $\mathbb{R}^n$ in einen höherdimensionalen reellen N-dimensionalen Vektorraum: $\mathbb{R}^N$. [CoVa 95]

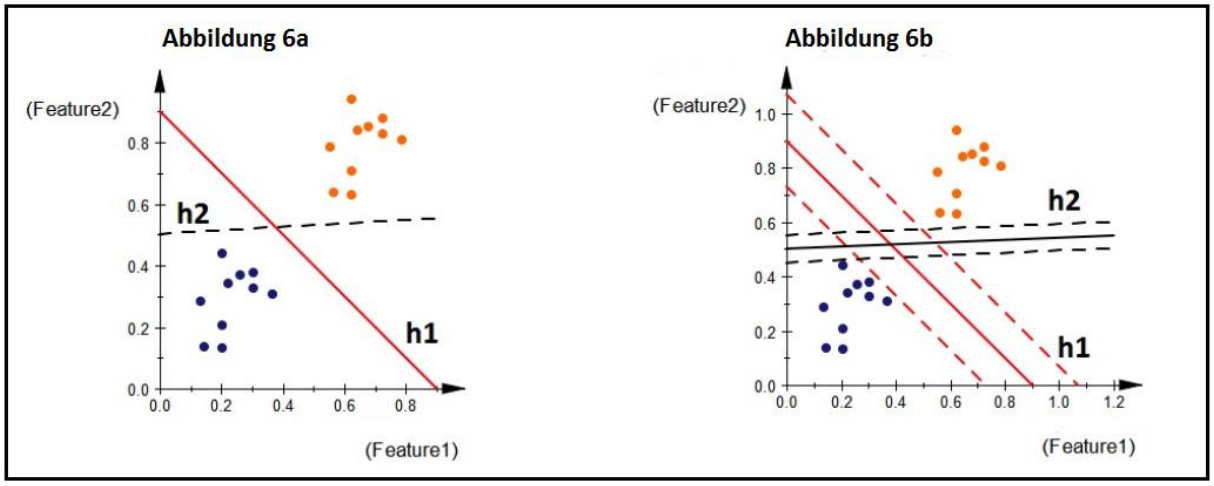


Abbildung 6: Visualisierung der Datenseparation mit SVM (modifiziert nach[LB 11])

(6a) Visualisierung von Daten in einem Vektorraum. Die eingezeichneten Geraden h1 und h2 separieren die Daten linear voneinander.

(6b) Anders als in Abbildung 6a sind die jeweiligen Abstände der Geraden h1 und h2 erkennbar. Die Gerade mit dem maximalen Abstand ist die gesuchte optimale Ebene.

Eingabevektorraum **Merkmalsvektorraum**

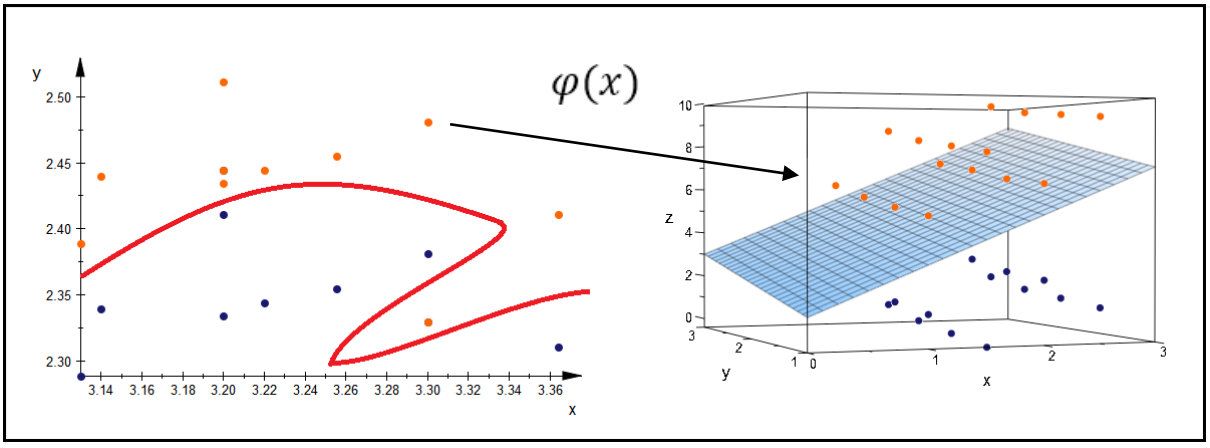


Abbildung 7: Lineare Separation von Daten durch Transformation in einen höherdimensionalen Raum. (modifiziert nach[LB 11])

Im Fall links (im Eingabevektorraum) ist es nicht möglich, die Daten von Patienten mit Lebertumoren (orangene Punkte) von den Patientendaten ohne Lebertumore (blaue Punkte) linear zu separieren. Aus diesem Grunde transformiert man die Datenpunkte mit Hilfe einer Transformationsfunktion: $\varphi(x)$, in einen höherdimensionalen Raum, um die Daten linear voneinander trennen zu können. [LB 11] Das Ergebnis sieht man im Fall rechts (im Merkmalsvektorraum).

Durch Hinzufügen von ausreichend Dimensionen lässt sich so jedes Problem in ein lineares Problem überführen. Diese Methode nennt man den Kerneltrick. Jedoch wird das Optimierungsproblem durch die Überführung der Datenmenge in einen höherdimensionalen Raum (Merkmalsvektorraum) deutlich schwieriger und algorithmisch rechenaufwendiger. Um im Merkmalsvektorraum die optimale Hyperebene, ohne Kenntnis der Transformationsfunktion: $\varphi(x)$ zu berechnen, benötigt man eine Kernelfunktion $K(x_i, y_i)$ mit deren Hilfe transformierte Vektoren skalar-multipliziert werden können. [CS RS 14].

$$K(x_i, y_i) \equiv \varphi(x_i) \cdot \varphi(y_i) \qquad (0)$$

[CS RS 14]

Durch das Skalarprodukt von zwei N-dimensionalen Vektoren würde sich algorithmisch die Komplexität der SVM auf $O(n) = 1$ beschränken.

In Anlehnung an [ChLin 11] gibt es folgende Kernelfunktionen:

Kerneltyp	Formel
linear:	$K(x_i, y_i) = x_i^T y_i$
polynomial:	$K(x_i, y_i) = (\gamma \cdot x_i^T y_i + r)^d ; \gamma > 0$
RBF:	$K(x_i, y_i) = e^{-\gamma \|x_i - y_i\|^2}; \gamma > 0$
sigmoid:	$K(x_i, y_i) = tanh(\gamma \cdot x_i^T y_i + r)$

Hierbei sind: "$d, r\ und\ \gamma$" die Kernelparameter der Trainingsdatensätze. Am häufigsten wird die RBF- Kernelfunktion genutzt. Die RBF-Kernelfunktion erinnert stark an die Gaußnormalverteilungsfunktion der Form:

$$f(\mu; \sigma \mid x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\left(\frac{(x-\mu)^2}{2\sigma^2}\right)} \qquad (1)$$

Genauer gesagt berechnet sich die Hyperebene, also die Entscheidungsfunktion des RBF-Kernels, mit folgender Formel:

$$f(x) = sign\left(\sum_{i=1}^{n} a_i \cdot e^{\left(\frac{|x-x_i|^2}{\sigma^2}\right)}\right) \qquad (2) \qquad [\text{CoVa 95}]$$

Es lässt sich eine starke Ähnlichkeit zwischen (1) und (2) feststellen. Dies liegt daran, dass sich die RBF-Kernelfunktion auf dem inversen Weg zur Gaußnormalverteilung herleiten lässt. (Siehe Mercer-Theorem in [CoVa 95])

2.3.3.2 Mathematische Beschreibung der SVM

In diesem Unterkapitel möchte ich auf die mathematische Modellierung der SVM bei linearer Separation von Daten eingehen. Sobald die Trainingsdaten linear voneinander zu trennen sind, existiert auch eine Hyperebene der folgenden Form:

$$H(z):\ \omega \cdot z + b = 0 \qquad (3)$$

[CoVa 95] [CS RS 14]

Dabei entspricht ω dem Normalenvektor, der orthogonal zur gesuchten Hyperebene $H(z)$ steht und auf diese zeigt. Der Koeffizient b gibt die Verschiebung der Hyperebene auf der Ordinate an. [CoVa 95] Mit diesem einfachen Trick aus der linearen Algebra können wir ω als Linearkombination der Stützvektoren (Support Vectors) ausdrücken:

$$\omega = \sum_{i=0}^{n} a_i z_i \qquad mit\ a_i = Gewichte\ vom\ iten\ Stützvektor\ z_i \qquad (4)$$

[CoVa 95]

Setzen wir nun die Linearkombination aus (4) in (3) ein, so erhalten wir die Entscheidungsfunktion:

$$H(z) = sign\left(\sum_{i=0}^{n} a_i z_i \cdot z + b\right) \qquad (5) \qquad [\text{CoVa 95}]$$

$z_i \cdot z$ ist hierbei das Skalarprodukt zwischen den Stützvektoren z_i und dem Vektor z im Merkmalsvektorraum. [CoVa 95] Nach Anwendung der Vorzeichenfunktion „$sign(x)$ “ definiert das resultierende Vorzeichen die Klassenzugehörigkeit und gibt somit an, wo sich die Vektoren relativ gesehen zur Hyperebene befinden. Somit differenzieren sich unsere Trainingsdaten mithilfe der Entscheidungsfunktion in zwei Klassen nämlich 1 und -1.
Die Vektoren der Klasse 1 haben ein positives Vorzeichen und liegen auf der positiven Seite der Hyperebene, die Vektoren der Klasse -1 haben ein negatives Vorzeichen und liegen auf der negativen Seite der Hyperebene siehe (Abbildung 8) [CS RS 14].
Beträgt der Wert des Vorzeichens gleich null, würden alle Vektoren der dazugehörigen Entscheidungsfunktion genau auf der Hyperebene liegen. [CS RS 14].
Zum besseren Verständnis können wir sagen die Klasse 1 entspricht den Patienten mit Lebertumoren und die Klasse -1 spiegelt die Patienten ohne Leberläsionen wieder.

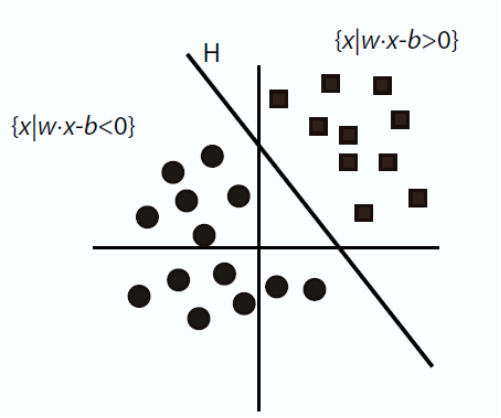


Abbildung 8: Lineare Separation von Samples
Die positiven und negativen Samples werden als schwarze Kreise und Quadrate visualisiert. Die Entscheidungsfunktion (Hyperebene) H trennt die Samples linear voneinander. [CS RS 14]

Der nächste Schritt besteht darin, die optimale Hyperebene zu finden, welche die gegebenen Datensamples, durch Definition eines Abstandes, am besten linear voneinander trennt. Hierfür stellen wir folgende Annahme in den Raum:

Eine Menge von abgebildeten Trainingsmustern (Merkmalspaaren):

$$\forall(x,y) \in \mathbb{R} \rightarrow \{x_1, y_1\}, \cdots\cdots \{x_i, y_i\} \qquad x_i \in \mathbb{R}^N \ \wedge \ y_i \in \{-1, 1\} \quad (6) \qquad [\text{CoVa 95}]$$

ist dann linear trennbar, sobald ein Vektor ω und ein Wert b valide sind für folgende Gleichungen:

$$H1\text{:} \ \ \omega \cdot x_i + b \geq 1, \quad wenn\, y_i = 1\,; \qquad H2\text{:} \ \ \omega \cdot x_i + b \leq -1, \qquad wenn\, y_i = -1\,;$$

$$y_i(x_i)\text{:} \quad \omega \cdot x_i + b \geq 1, \qquad \forall i = 1 \ldots\ldots L \qquad (7) \qquad [\text{CoVa 95}]$$

Der Abstand der optimalen Hyperebene lässt sich aus folgender Formel berechnen:

$$\rho(\omega, b) = \min_{|x,y=1|} \frac{x \cdot \omega}{|\omega|} - \max_{|x,y=-1|} \frac{x \cdot \omega}{|\omega|} \qquad (8) \qquad [\text{CoVa 95}]$$

Das „b“ kürzt sich hier aus der Gleichung heraus. Dabei entspricht „min“ der oberen Grenze und „max“ der unteren Grenze des Abstandes.

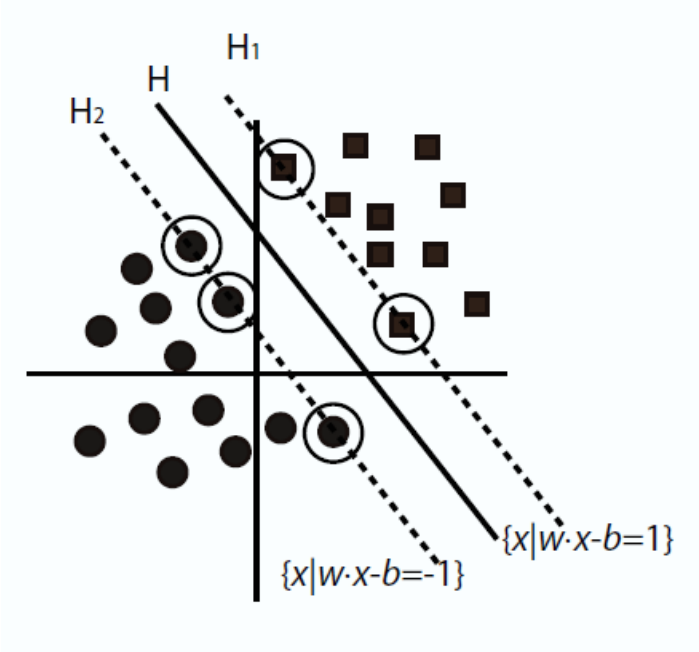
Vektoren werden dann als Stützvektoren bezeichnet, wenn folgendes gilt:

$$y_i(x_i)\text{:} \quad \omega \cdot x_i + b = \pm 1 \qquad (9) \qquad [\text{CoVa 95}]$$

Den oben beschriebenen Zusammenhang kann man mit folgender Abbildung illustrieren:

Abbildung 9: Bestimmung der optimalen Hyperebene durch Einführung von Margins

Die optimale Hyperebene H liegt genau in der Mitte zwischen der Ebene H1 und H2.
Die Vektoren, die direkt auf den Ebenen H1 und H2, liegen werden als Stützvektoren (Support Vectors) bezeichnet. [CS RS 14]



2.3.3.3 SVM-Parameter

Im diesem Grundlagenkapitel möchte ich ein allgemeines Verständnis für den Einsatz der SVM-Parameter aufbauen. Um SVMs richtig einsetzen zu können, muss man sich bewusst darüber sein, welche Kombination der SVM-Parameter für welches Problem am besten geeignet ist. Grundsätzlich gibt es folgende Parameter, die bei der Verwendung einer SVM benutzt und variiert werden können: (siehe Abbildung 10)

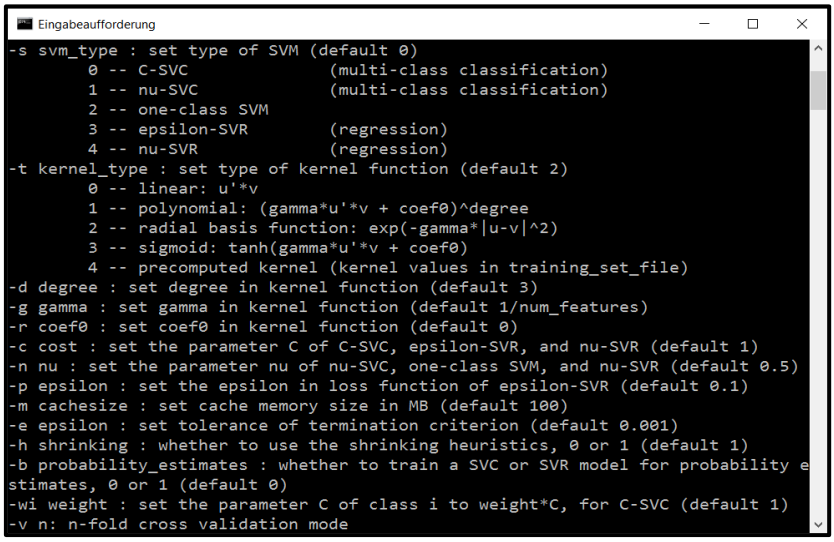


Abbildung 10: SVM-Parameter

Ich möchte im nächsten Schritt illustrieren, welchen Effekt eine Variation der SVM-Parameter auf die Bildung der Hyperebene, des Margins und der Entscheidungsfunktion haben kann. Die SVM-Parameter, die in dieser Thesis am meisten verwendet wurden sind: Der RBF-Kernel (Parameter t 2), der Gammawert (Parameter g), der Kostenparameter (Parameter C) und der multimodale SVM-Typ (Parameter s 0). Mit Verwendung des Kostenparameters kann man den Margin beeinflussen. (siehe Abbildung 11)

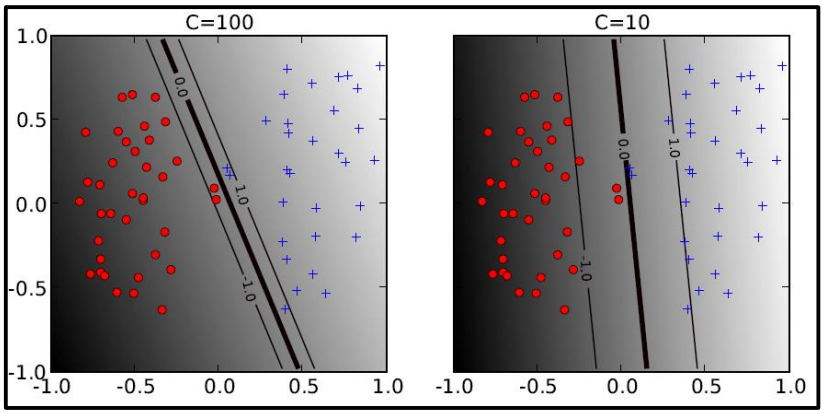


Abbildung 11: Einfluss des Kostenparameters auf den Margin [AB JW 09]

Die roten und blauen Datasamples in Abbildung 11 werden durch eine Hyperebene (dicke schwarze Linie) linear voneinander getrennt. Die gestrichelten Linien sind der jeweilige Abstand (Margin).

Das Verhältnis zwischen dem Margin und den Kosten „C“ ist umgekehrt proportional. D.h. bei einem kleinen Kostenwert wird der Margin größer und die bounded support vectors (also die support vectors, die zwischen dem Margin und der Entscheidungsgrenze (Hyperebene) liegen), können ignoriert werden. (siehe Abbildung 11 rechts). Für den Falll, dass der Kostenwert C stark erhöht ist, wird der Margin kleiner. (siehe Abbildung 11 links) Diejenigen Stützvektoren, die in diesem Fall zwischen der Hyperebene und dem Margin liegen, werden zu so genannten Abstandsfehlern (engl. margin errors). [AB JW 09] Auch die Kernelparameter (Abbildung 10 t-Parameter) nehmen bei der Bildung der Entscheidungsgrenze (Hyperebene) der SVM eine bedeutsame Rolle ein. Der Parameter-Grad (degree d) des polynomial-Kernels und der Gammawert (Parameter g) des RBF-Gaußkernels beeinflussen die Flexibelität der SVM bei Berechnung der Hyperebene. (siehe Abbildung 12 und 13) [AB JW 09] In Abbildung 12 ist ersichtlich, dass ein höherer Grad im polynomial-Kernel eine flexiblere Hyperebene berechnet.

$$Poly(x_i, y_i) = \left(g \cdot x_i^T y_i + r\right)^d \;;\; g > 0 \quad (aus\ Kapitel\ 2.3.3.1)$$

So ist ein Polynomgrad von 1 äquivalent mit der linearen Kernelfunktion, weswegen die Entscheidungsgrenze auch eine triviale lineare Funktion ist. (Abbildung 12 links) Würde der Grad bei 2 liegen, ist die Entscheidungsgrenze parabolisch geformt, da wir eine Polynomfunktion zweiten Grades haben. (Abbildung 12 mitte) Bei einem Grad von 5 krümmt sich die Entscheidungsgrenze noch stärker und trennt somit die roten Datensamples besser von den blauen Samples ab. (Abbildung 12 rechts)

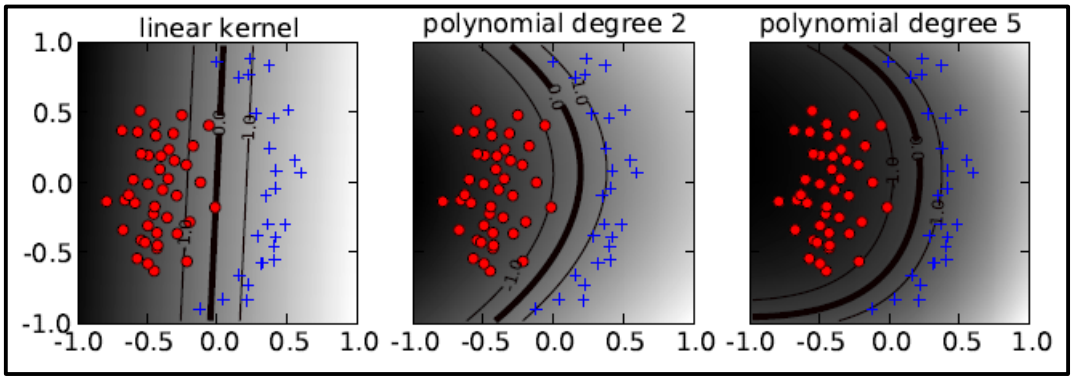


Abbildung 12: Einfluss vom polynomial-Kernel bei verschiedenen Graden der Polynomkernelfunktion [AB JW 09]

Schauen wir uns jetzt den Einfluss des Gammaparameters beim RBF-Kernel an:

$$K(x_i, y_i) = e^{-g\|x_i - y_i\|^2};\ g > 0 \quad (aus\ Kapitel\ 2.3.3.1)$$

Der Gammaparameter g gibt bei der Gaußkernelfunktion die Breite der Gaußglocke an. Abbildung 13 zeigt dass, wenn der Gammawert sehr klein ist (Fall links), die Entscheidungsgrenze fast linear ist. Genau wie beim Grad-Parameter der Kernelpolynomfunktion sorgt eine Erhöhung des Gammawertes dafür, dass die Flexibilität der Entscheidungsgrenze auch erhöht wird. (siehe Abbildung 13 mitte) Ist der Gammawert groß, führt dies zum sogenannten Overfitting (Abbildung 13 rechts). [AB JW 09] Je größer der Gammawert, desto stärker ist die Krümmung der Hyperebene.

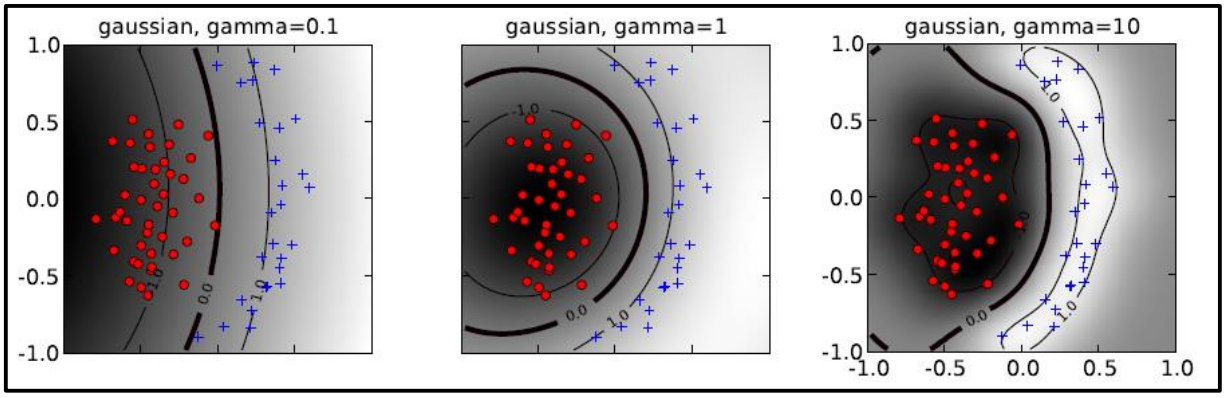


Abbildung 13: Einfluss verschiedener Gammawerte auf die Entscheidungsgrenze beim RBF-Kernel [AB JW 09]

Ein Overfitting bedeutet, dass man dem Klassifikator zu detailreiche Informationen liefert, sodass der Klassifikator auswendig lernt. Zusammengefasst kann man für dieses Kapitel sagen, dass man anhand der Beispiele gesehen hat, wie sich eine Änderung der Parameter auf die Bildung der Entscheidungsgrenze und somit auf die Klassifikation auswirkt. Die Kernelparameter (Gamma und Grad) haben eine essentielle Wirkung auf die Krümmung der Hyperebene. Oft werden der RBF-Gaußkernel und der polynomial-Kernel bei Verwendung einer SVM gewählt, da diese eine hohe Flexibilität bei der Bildung der Entscheidungsgrenze haben. Aber die Gefahr für ein Overfitting ist aufgrund der Flexiblität groß. [AB JW 09]

2.4 Verwendete Software

Für die Implementierung der Support Vector Machines wurde die Bibliothek LIBSVM verwendet und in das Framework MITK integriert.

2.4.1 LIBSVM

Nach einer ausführlichen Beschäftigung mit den Grundlagen der SVM, komme ich nun zu der Bibliothek LIBSVM. LIBSVM steht für engl. Library for Support Vector Machines) und ist somit eine Bibliothek für SVMs, die in C /C++ geschrieben ist. Zudem hat die LIBSVM Interfaces zu den Programmiersprachen: Java, Phyton etc. Die LIBSVM hat zum Ziel, dass SVMs mit möglichst wenig Aufwand in Anwendungen eingebettet werden können. [ChLin 11]

Folgende Aufgaben unterstützt die LIBSVM:

- **SVC = Support Vector Classifikation (für zwei oder mehrere Klassen)**
- SVR = Support Vector Regression
- One-class SVM [ChLin 11]

In der vorliegenden Bachelorthesis wurde die SVC genutzt, da wir mit multimodalen Bildern arbeiten, und wir somit zwischen zwei Klassen differenzieren wollen, nämlich Lebertumor und gesundes Lebergewebe. Typischerweise wird die LIBSVM hauptsächlich für folgende zwei Schritte genutzt:

1. Trainiere auf einem Datensatz, um ein Modell zu erstellen
2. Verwende das Modell, um einen Teil des Testdatensatzes vorherzusagen

[ChLin 11]

Die LIBSVM liefert ein Programm mit, das eine Funktion hat um die Parameter der LIBSVM zu validieren. Für die Parameteroptimierung wurde das Kreuzvalidierungsverfahren engl. cross-validation verwendet, welches eine cross-validation accuracy (CVA) liefert.

	Wahre Bedingung	
Vorausgesagte Bedingung	Patientenleber mit Tumor	Patientenleber ohne Tumor
Patientenleber mit Tumor	True positive TP	False positive FP
Patientenleber ohne Tumor	False negative FN	True negative TN

Abbildung 14 : Vierfeldertafel

$$TP = richtig\ klassifizierter\ Tumor, TN = richtig\ klassifizierte\ Leber\ ohne\ Tumor,$$
$$FP = falsch\ klassifizierte\ Leber\ ohne\ Tumor, FN = falsch\ klassifizierter\ Tumor$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{Anzahl\ der\ richtig\ klassifizierten\ Fälle}{Gesamtanzahl\ aller\ klassifizierten\ Fälle}$$

Je höher die Accuracy ist, desto besser ist die optimale Parameterkonfiguration der LIBSVM.

2.4.2 MITK

Das Tool MITK steht für engl. Medical Imaging Interaction Toolkit und ist eine Klassenbibliothek, die auf den Erweiterungen ITK (engl. Insight Segmentation and Registration Toolkit) und VTK (engl. Visualization Toolkit) basiert. [IW VI 04] MITK ist in C++ geschrieben und ein Open Source Produkt. Ziel von MITK ist es, die Erschaffung von klinisch nutzbarer bildgebender Software zu erleichtern. [IW VI 04] Dabei unterstützt ITK Segmentierungsalgorithmen wie z.B. Region Growing, VTK hat die Fähigkeit einer leistungsstarken Visualisierung der medizinischen Bilder. Mit MITK ist es möglich, medizinische Bilder verschiedenster Modalitäten einfach anzusehen, indem die Bilder in den Datamanager per drag and drop hineingezogen werden können (siehe Abbildung 15). Nun ist das geladene Bild auch in den vier Render -Windows sichtbar und kann mithilfe der Radiobuttons aktiviert oder deaktiviert werden. Es ist auch möglich mehrere Bilder übereinanderzulegen, wie dies in (Abbildung 15) ersichtlich ist, um beispielsweise den Tumorbereich vom Nichttumorbereich der Leber zu differenzieren. Der Transparenzgrad der Bilder kann auch variiert werden.

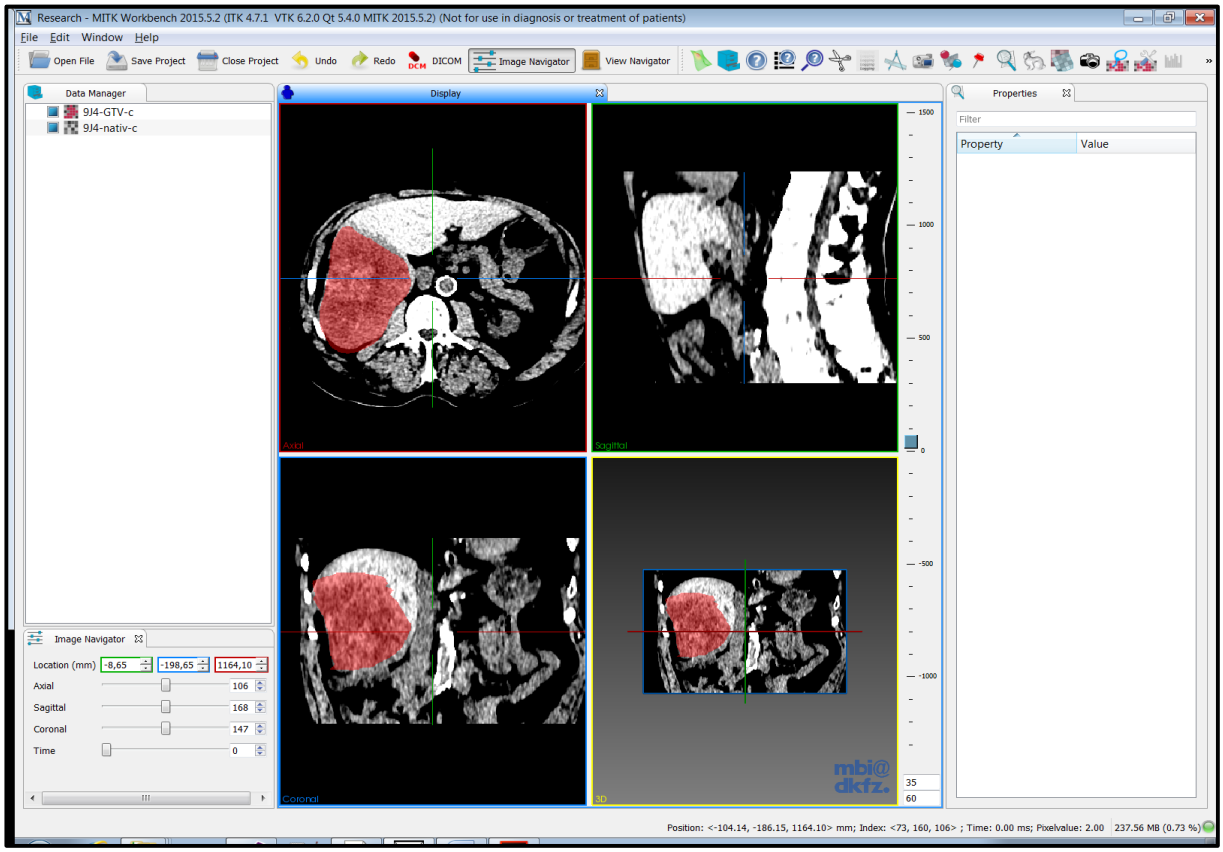


Abbildung 15: Grafische Benutzeroberfläche von MITK

Links zu sehen ist der Datamanager, in der Mitte befinden sich die vier Render-Windows. Drei von diesen Windows illustrieren die drei orthogonalen Schnitte 2D durch den Körper (transversal (oben links), sagittal (oben rechts) und coronar (unten links)). Das vierte Fenster rechts unten ist eine 3D-Ansicht aller Ebenen.Im rechten Bereich der GUI sehen wir das Propertymenü, alternativ kann dort auch das Segmentierungsmenü hinzugefügt werden.

3. Entwurf und Implementierung

In diesem Kapitel geht es darum, wie die SVM zur Lebertumorklassifikation entworfen und implementiert wurde. Die entwickelte SVM trägt den Namen „CLSVMClassification“ und nutzt eine Wrapperklaasse „LibSVMClassifier“, um die LIBSVM in die Anwendung zu integrieren.

Für die Implementierung der SVM wurde die Programmiersprache C++ verwendet. Die Software wurde mithilfe der Entwicklungsumgebung Microsoft Visual Studio 2013 realisiert. Dabei wurde auch die Klassenbibliothek Qt5 in Visual Studio 2013 integriert, damit die Programmiersprache C++ um einige Fähigkeiten bereichert wird. Die entwickelte LIBSVM wurde anhand einer selbstimplementierten Testklasse „mitkLibSVMClassifierTest.cpp“ validiert. (siehe Kapitel 4). Um die Qualität des Klassifikationsergebnisses der entwickelten LIBSVM zu verbessern, wurden zusätzliche Features Kapitel 5.1.2 integriert. Für das Training und die Prädiktion stellt die benutzte Wrapperklasse „mitkLibSVMClassifier.cpp“ zwei Methoden bereit, die von der entwickelten LIBSVM Anwendung genutzt werden.

In den folgenden Unterkapiteln wird zunächst beschrieben, wie ein Klassifikationsprozess bei einer automatischen Lebertumorsegmentierung abläuft. (Kapitel 3.1) Danach folgt eine Beschreibung des Softwareentwurfs der entwickelten SVM (Kapitel 3.2) mit den daran enthaltenen Softwareanforderungen, welche die SVM erfüllen sollte. Abschließend wird darauf eingegangen, wie genau die defnierten Softwareanforderungen an die SVM umgesetzt worden sind (Kapitel 3.3), d.h: wie die entwickelte SVM implementiert wurde.

3.1 Ablauf eines Klassifikationsprozesses bei einer automatischen Lebertumorsegmentierung

Grundsätzlich lässt sich der Klassifikationsprozess der entwickelten SVM in folgende Schritte differenzieren:

1) Vorbereitung der Trainingsdaten für das Training
2) Das Training
3) Vorbereitung der Testdaten für das Prädizieren
4) Das Prädizieren

Schritt 1: Vorbereitung der Trainingsdaten für das Training:

In diesem Schritt geht es darum, dass man wissen will, wie die medizinischen Bilddaten in die entwickelte SVM-Anwendung kommen. Zuerst werden die medizinischen Bilder aus einer Datenbank (wie z.B. XNAT) in einem DICOM Format zur Verfügung gestellt. (siehe Abbildung 18). DICOM ist ein Standard für die Speicherung und den Austausch von medizinischen Bilddaten. Des Weiteren wird jedes Bild mit Angabe des Dateipfades und der Modalität in eine XML-Datei nach folgendem Format integriert: (siehe Abbildung 16)

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!-- Beispiel XML für Datenintegration-->
<col name="Images" >
  <subcol name="PatientA" description="Dummy Layer" id="PatientA" >
    <data name="Alex" description="--" id="Alex" >
      <item name="nativ" filepath="" link="PatientA/Alex/Alex-nativ.nrrd" />
      <item name="ven" filepath="" link="PatientA/Alex/Alex-ven.nrrd" />
      <item name="art" filepath="" link="PatientA/Alex/Alex-art.nrrd" />
    </data>
  </subcol>
  <subcol name="PatientB" description="Dummy Layer" id="PatientB" >
    <data name="Bastian" description="--" id="Bastian" >
      <item name="nativ" filepath="" link="PatientB/Bastian/Bastian-nativ.nrrd" />
      <item name="ven" filepath="" link="PatientB/Bastian/Bastian-ven.nrrd" />
      <item name="art" filepath="" link="PatientB/Bastian/Bastian-art.nrrd" />
    </data>
  </subcol>
</col>
```

Abbildung 16: Aufbau einer XML-Datei

Jedes Element item hat ein Attribut name, in dem die jeweilige Modalität gespeichert ist (z.B ven = venös) und ein Attribut link in dem der passende Pfad zur Bilddatei hinterlegt ist. (z.B PatientA/Alex/-ven.nrrd)

In XML arbeitet man mit Starttags (<example>) und Endtags (</example>). Sehr wichtig ist auch die eingerückte Struktur der Tags, wie in (Abbildung 16) ersichtlich. Die Groß- und Kleinschreibung der Tags ist im XML-Standard zu vernachlässigen. Werden den jeweiligen Tags Werte zugewiesen, so nennt man diesen Wert ein Attribut. Ist hingegen dem Attribut ein Wert zugeteilt, spricht man von einem Attributwert. Beispiel aus (Abbildung 16):

<item name = „ven“ ..>
<Element Attribut = „Attributwert“ ..>

Die XML-Struktur kann man in eine Baumstruktur der folgenden Form konvertieren:

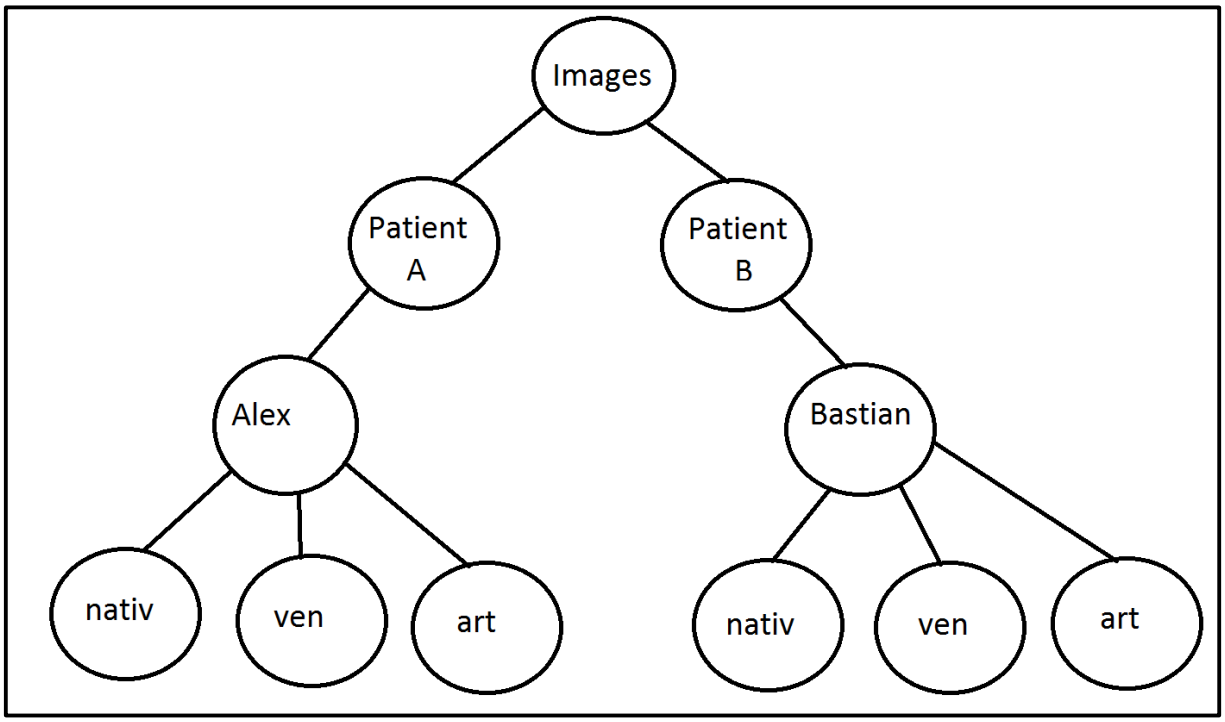


Abbildung 17: XML Datei als Baumstruktur

XML-Baumstruktur generiert anhand der eingerückten Tags. Der Tag <col> ist nicht eingerückt. Somit steht der Wert im Tag <col> „Images“ als Wurzelknoten ganz oben in dem generierten Baum. Die Elternknoten des Baumes (Patient A und Patient B) sind in der XML-Struktur aus (Abbildung 16) der nächste eingerückte Tag, hier also <subcol>. Die Kindelemente des Baumes (Alex und Bastian) sind durch die nächste Einrückung des Tags <data> entstanden. Zuletzt sind die Blattknoten (nativ, ven, art) zu sehen, diese gehören zur den am stärksten eingerückten Tags <item>.

Die Baumtiefe des abgeleiteten XML-Dokumentes bestimmt sich aus der Anzahl der Einrückungen für das jeweilige Element. Zur Integration des DICOM-Bildes in die Textdatei werden die jeweiligen Attributwerte in den <item> Tags schematisch angelegt. Gemäß (Abbildung 18) werden die Bilddaten mithilfe der SVM aus der XML und Textdatei gelesen und in folgendes Matrixformat konvertiert:

Codeauszug1 der implementierten SVM:

```
 if (doTraining)
      {
auto trainDataX = mitk::DCUtilities::DC3dDToMatrixXd(trainCollection, mo-
dalities, trainMask);
auto trainDataY = mitk::DCUtilities::DC3dDToMatrixXi(trainCollection,
trainMask, trainMask);
```

Sourcecode in Visual Studio mit der Programmiersprache C++ geschrieben.

Strukturierung des Sourcecodes in der Matrixform:

$$VectorY = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 2 \end{pmatrix} \qquad MatrixX = \begin{pmatrix} 200 & 61 & 111 \\ 30 & 70 & 99 \\ 50 & 53 & 4 \\ 32 & 3 & 117 \end{pmatrix} \qquad (11)$$

Dabei entspricht der Inhalt des Objektes `trainDataY` dem Labelvektor „VectorY“. (siehe Formel 11) Der Wert 1 wird gesundem Lebergewebe, der Wert 2 wird Lebertumorgewebe zugeordnet. Die „MatrixX“ aus (Formel 11) wird im Objekt `trainDataX` (siehe Codeauszug1) gespeichert. Dabei entspricht jeder Spaltenvektor mit seinen Voxelelementen (z.B. $(200, 30{,}50{,}32)^T$) einem kompletten Grauwertbild oder einer Modalität. Diese Daten werden nun der SVM-Anwendung für den zweiten Schritt (das Training) bereitgestellt.

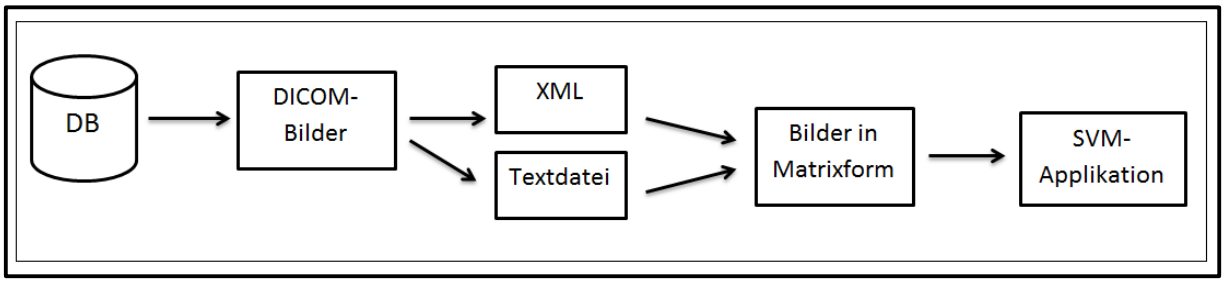


Abbildung 18: Datenvorbereitungspipeline eines Klassifikationsprozesses

Enthält die wesentlichen Schritte, die notwendig sind, um die medizinischen Bilddaten für die Klassifikation von Lebertumoren vorzubereiten.

Schritt 2: Das Training

Der Prozessablauf des Trainings der SVM für eine Lebertumorsegmentierung kann mithilfe von (Abbildung 19) schrittweise erläutert werden: Die in (Abbildung 19) ersichtliche Trainingsmatrix setzt sich aus der Kombination der partiellen Matrizen `trainDataX` und `trainDataY` zusammen.(siehe Codeauszug1) Die Vektorelemente des ersten Spaltenvektors $(1, 2, 1, 1, 2, 2)^T$ aus der Trainingsmatrix sind äquivalent zu den Voxelelementen des Labelbilds. (siehe Abbildung 19) Das nachfolgende Grauwertbild mit seinen Voxelelementen wird somit den Vektorelementen des zweiten Spaltenvektos der Trainingsmatrix zugewiesen. Im dritten Spaltenvektor der Trainingsmatrix wird das kantendetektierte Texturbild zugewiesen. Alle weiteren Bilder werden als zusätzliche Spaltenvektoren in der Trainingsmatrix gespeichert. Bevor es mit dem eigentlichen Training losgeht, werden die Parameter der SVM gesetzt und optimiert. Ein gängiges Verfahren, um die Parameter zu optimieren, ist das Kreuzvalidierungsverfahren, welches ich im (Kapitel 4.3) erläutern möchte. Nach erfolgreicher Parameteroptimierung kann die SVM-Anwendung gestartet werden und erstellt als Ergebnis des Trainingsprozesses eine Model-Datei. (siehe Abbildung 20)

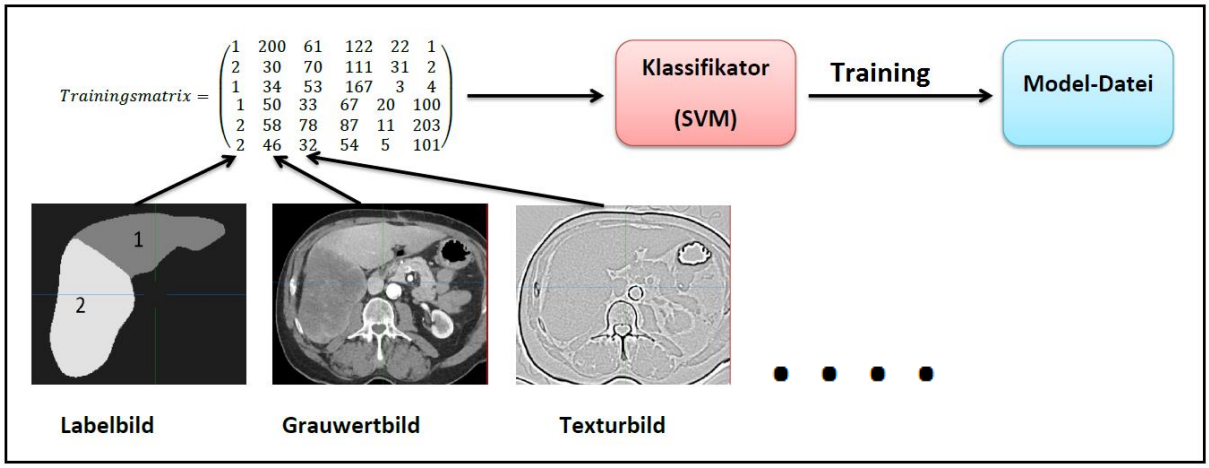


Abbildung 19: Prozessablauf für das Training der SVM

```
svm_type c_svc
kernel_type rbf
gamma 0.0006
nr_class 2
total_sv 18
rho 0.154782
label 1 2
nr_sv 9 9
SV
0.2305077394003501 1:146 2:68 3:101
0.3302504254028029 1:122 2:58 3:134
0.4489101963758092 1:142 2:80 3:102
0.09584096775551353 1:151 2:63 3:114
0.447373147433417 1:140 2:65 3:135
0.2976930276317124 1:49 2:59 3:119
0.02719726012045989 1:81 2:92 3:109
0.93657086829259 1:84 2:62 3:88
1.874614603146205 1:65 2:61 3:94
```

Abbildung: 20 Beispiel einer Model-Datei

Die nach dem Training generierte Model-Datei ist nach folgendem Prinzip aufgebaut:
Zuerst sieht man die eingestellten SVM Parameter, die nicht standardmäßig gesetzt werden. (hier: SVM-Typ, Kernelfunktion, Kernelparameter), gefolgt von der Anzahl der definierten Klassen (nr_classes). Mit total_sv wird die Anzahl der Supportvektoren aller Klassen bezeichnet. SV listet die Supportvektoren auf.

Schritt 3: Vorbereitung der Testdaten für das Prädizieren

In diesem Schritt gehen wir analog vor, wie aus (Abbildung 18) ersichtlich. Der einzige Unterschied besteht darin, dass der Testdatensatz in eine Testmatrix der folgenden Form eingelesen wird:

Codeauszug2:

```
auto testDataX =
mitk::DCUtilities::DC3dDToMatrixXd(testCollection,modalities, testMask);
```

Strukturierung des Sourcecodes in der Matrixform:

$$Testmatrix := \begin{pmatrix} 122 & 34 & 99 \\ 24 & 87 & 42 \\ 4 & 203 & 1 \end{pmatrix} \qquad (12)$$

Dabei entspricht das Objekt `testDataX` dem Inhalt der Testmatrix aus Formel 12. Die Matrix (Formel 12) ist ähnlich aufgebaut wie die Matrix in (Formel 11). Das bedeutet, der erste Spaltenvektor entspricht dem ersten eingelesenen Bilddatensatz, der zweite Spaltenvektor entspricht den Voxeln des nächsten eingelesenen Bildes im Testdatensatz etc. Auch hier werden die Testdaten der SVM-Applikation bereitgestellt, insbesondere für den vierten Schritt das Prädizieren.

Schritt 4: Das Prädizieren

Im diesem Schritt erfolgt die Klassifikation durch die SVM. Hierfür wird die generierte Model-Datei aus dem Schritt 2 und die Testdatenmatrix aus dem Schritt 3 dazu verwendet, um den Tumor für diesen Testdatensatz zu prädizieren. (siehe Abbildung 21) Das Ergebnis der Prädiktion ist eine komplette Tumorsegmentierung der Leber. (siehe Abbildung 21 Ergebnissegmentierung)

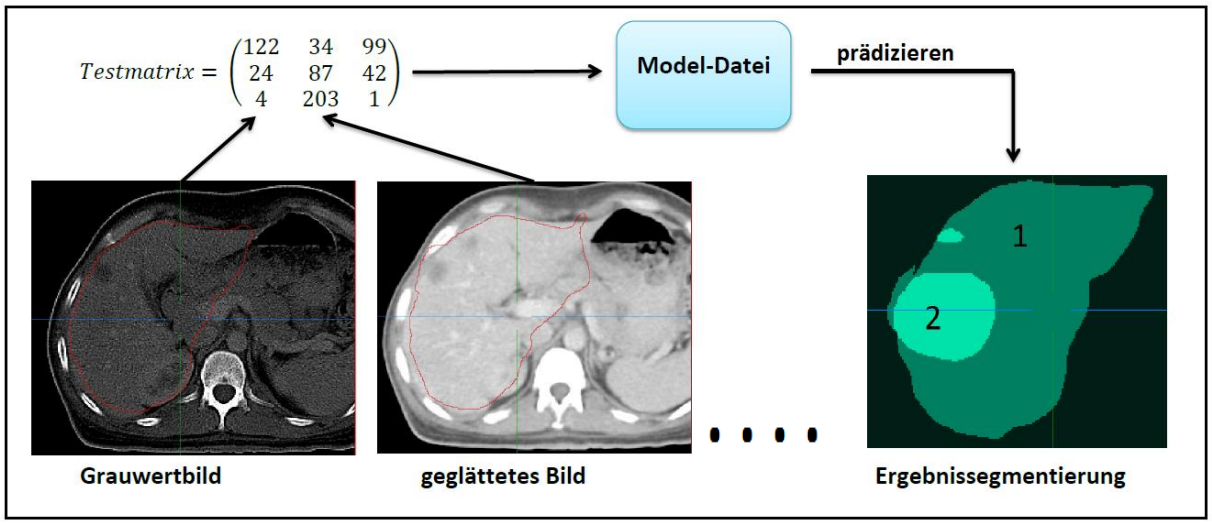


Abbildung 21: Prozessablauf für das Prädizieren der SVM

3.2 Softwareentwurf zur SVM-Klassifikation

Die integrierte SVM (CLSVMClassification.cpp) wurde nach der im Folgenden beschriebenen Klassenarchitektur entworfen. (siehe Abbildung 22) Zur Kapselung der LIBSVM wurde eine Wrapperklasse „LibSVMClassifier.cpp“ genutzt, die wichtige Methoden für das Training und das Prädizieren bereitstellt. Somit kann die entwickelte SVM auch für andere Anwendungen verwendet werden. Zudem kann die implementierte SVM für alle Tumorarten in CT-Bildern zur Klassifikation eingesetzt werden. Sie bietet auch I/O-Methoden (Input/Output) aus dem Modul „MITKCORE_EXPORTATION“, um die Trainings- und Testdaten in die bestehende Applikation zu laden und das trainierte Modell sowie die prädizierten Ergebnisbilder zu speichern. Das selbst entwickelte Programm „mitkLibSVMTestSuite.cpp“ wird zum Test der entwickelten SVM verwendet. Mit dem eigens implementierten Programm „CLAddImages“ können binäre Labelbilder generiert werden, welche essentiell für den Klassifikationsprozess der entwickelten SVM-Anwendung sind. Sollen zusätzliche Features, zur Verbesserung des Klassifikationsergebnisses generiert werden, kann das entwickelte Programm vom DKFZ „CLVoxelFeatures“ genutzt werden.

Folgende Anforderungen werden an die entwickelte SVM gestellt:

a) Die Software soll mithilfe der SVM so implementiert werden, dass beliebig viele medizinische digitale Bilder verschiedenster Modalitäten eingelesen und diese in eine Trainingsdatenmatrix und eine Testdatenmatrix konvertiert werden. Dieser Schritt ist notwendig, um später die SVM-Parameter anhand der externen LIBSVM zu optimieren und zu validieren.
b) Die Software soll es aufgrund des Trainings ermöglichen, entsprechende Modeldateien zu generieren und diese in einem Verzeichnis zu persistieren. Durch die Persistierung des Models ist es möglich, direkt zum nächsten Schritt der Klassifikation zu gehen, das Prädizieren. Zudem können wir das SVM-Modell mit dem Modell der externen LIBSVM vergleichen.
c) Die Software soll eine Wrapperklasse beinhalten, welche in der Lage ist, die LIBSVM in die bestehende Applikation zu integrieren.
d) Die Software soll mithilfe einer gegebenen Modeldatei und eines ungelabelten Bildes Lebertumore in diesem gegebenen Bild effektiv klassifizieren und vollautomatisch segmentieren. Diese Anforderung ist sehr essentiell für die Mediziner, damit diese den Lebertumor nicht mehr von Hand Schicht für Schicht segmentieren müssen.

e) Für die entwickelte Software sollen entsprechende Testklassen geschrieben werden, die CSV-Dateien in einem bestimmten Format einlesen und in Matrizenpaare konvertieren. Anschließend wird mit den Datensätzen eine Modeldatei trainiert und ein Tumorlabelvektor prädiziert. Als Ergebnis soll der Test eine Accuracy liefern, inwiefern der prädizierte Labelvektor mit dem tatsächlichen Spaltenvektor der Matrix übereinstimmt. Diese Testklassen testen, ob das Trainieren und Prädizieren der SVM richtig funktioniert.

f) Die Software soll auch zusätzliche Features, wie z.B. Texturbilder durch Gaußglättung, einlesen und mit diesen Features die SVM trainieren und prädizieren. Durch die Hinzunahme von zusätzlichen Features soll das Klassifikatonsergebnis wesentlich verbessert werden, sodass eventuelles Rauschen beseitigt oder zwischen den Tumorvoxeln besser interpoliert wird.

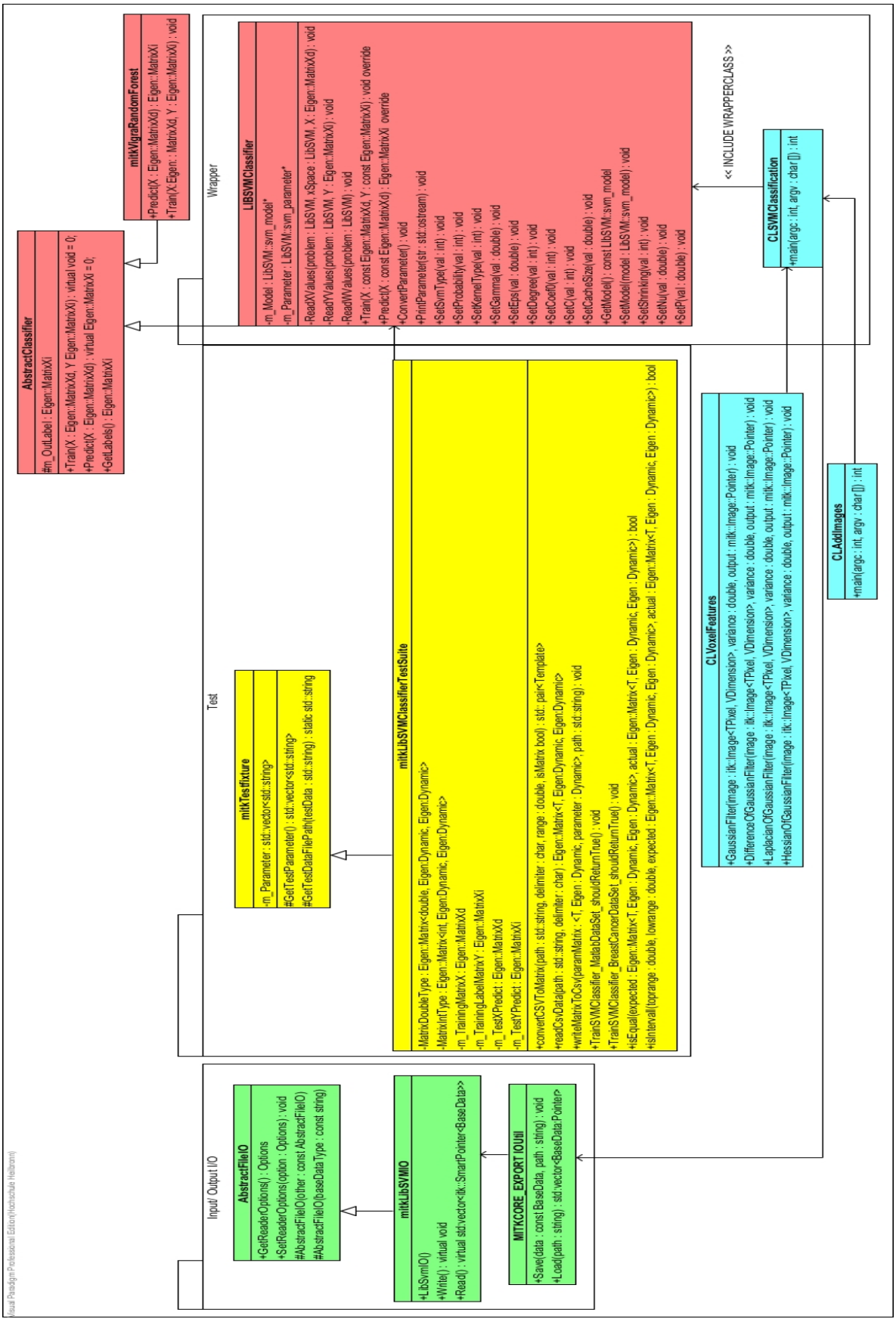


Abbildung 22: Klassendiagramm zur entwickelten LIBSVM

3.3 Implementierung von SVM in MITK

Im folgenden Unterkapitel geht es um die Umsetzung der im Kapitel 3.2 definierten Anforderungen. Für die Integration der SVM in MITK wurde das Programm „CLS-VMClassification.cpp“ implementiert. Die in Kapitel 3.2 definierten Anforderungen a) bis f) wurden wie folgt umgesetzt:

a) Die entwickelte Anwendung kann beliebig viele medizinische digitale Bilder (Abbildung 23) mithilfe eines Collectionreaders und einer zur Verfügung gestellten XML-Collection-Datei einlesen (Kapitel A 2), in welcher wie in Kapitel 3.1 beschrieben die Bilder XML-spezifisch hinterlegt werden können. Parallel dazu sind die eingelesenen Bilder der jeweiligen Patienten in einer Ordnerverzeichnisstruktur hinterlegt. Die verwendeten Bildmodalitäten (art, ven und nativ) werden in der XML-Datei und in zwei Textdateien referenziert. Dabei sind die Textdateien (Abbildung 24) Inputparameter des Kommandozeilenprogrammes „CLSVM-Classification.cpp“. In der Textdatei aus Abbildung 24 links werden die SVM-Parameter initialisiert sowie die Trainings- und Testdatensätze übergeben. In der rechten Eingabedatei aus Abbildung 24 werden die Patientenkollektive, die jeweils aus einer Trainingsgruppe und einer Testgruppe bestehen, festgelegt. Desweiteren werden die Bildmodalitäten gesetzt, die die SVM einlesen soll. Mit Hilfe des Codeauszug: 1 aus Kapitel 3.1 werden die eingelesenen Bilddaten in Trainingsmatrizen konvertiert (siehe Kapitel 3.1 Schritt 1). Aus Codeauszug 2 (in Kapitel 3.1) geht die Konvertierung der Bilddaten in die Testmatrix hervor. Desweiteren nutzt das entwickelte Programm Methoden, um die Trainings- und Testmatrizen in separate CSV-Dateien herauszuschreiben. Dieser Vorgang ist für das mitgelieferte Programm der entwickelten LIBSVM notwendig, um die SVM-Parameter zu optimieren.

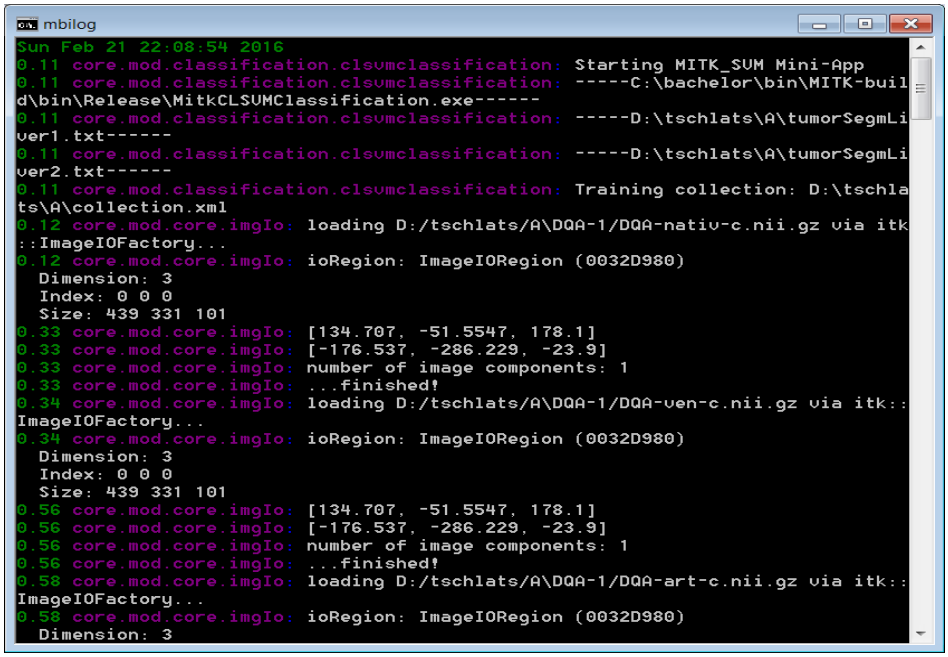


Abbildung 23: Einlesevorgang der Bilddaten mit der entwickelten SVM

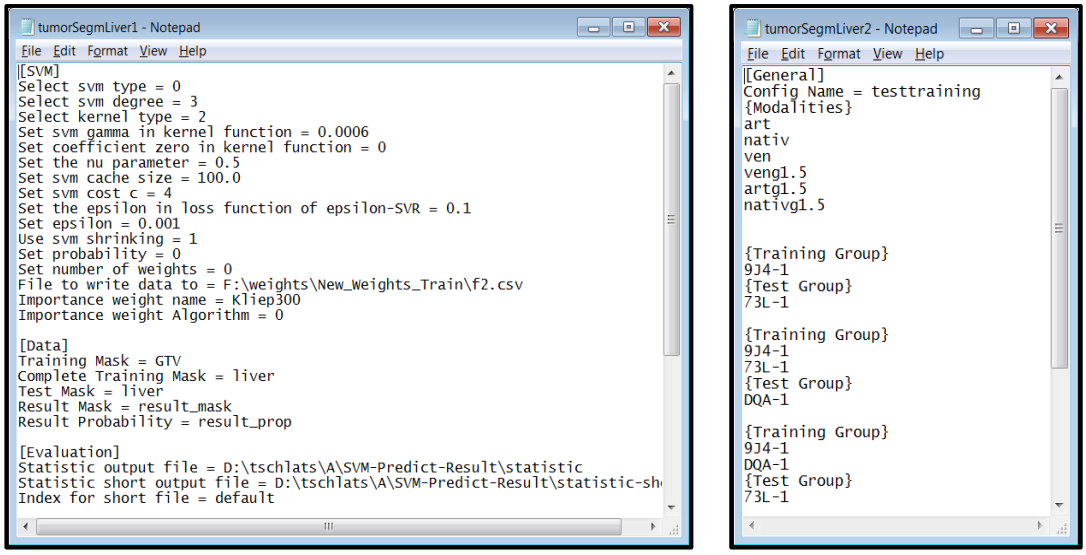


Abbildung 24: Inputparameterdateien für die entwickelte SVM

b) Das implementierte Programm „CLSVMClassification.cpp“ nutzt die Train-Methode der inkludierten Wrapperklasse „LibSVMClassifier.cpp“, um nach dem Training eine Model-Datei zu erstellen. (für Trainingsablauf siehe Kapitel 3.1 Schritt 2) Diese wird dann unter Verwendung der I/O Save Methode des Modules „MITKCORE_EXPORTATION“ in einem Verzeichnis „SVM-Train-Result“ gespeichert. (siehe Abbildung 25) Dabei sieht man in Abbildung 25, welche SVM-Parameter wie gesetzt worden sind. Unter den gesetzten Parametern sieht man das während des Trainings berechnete Modell.

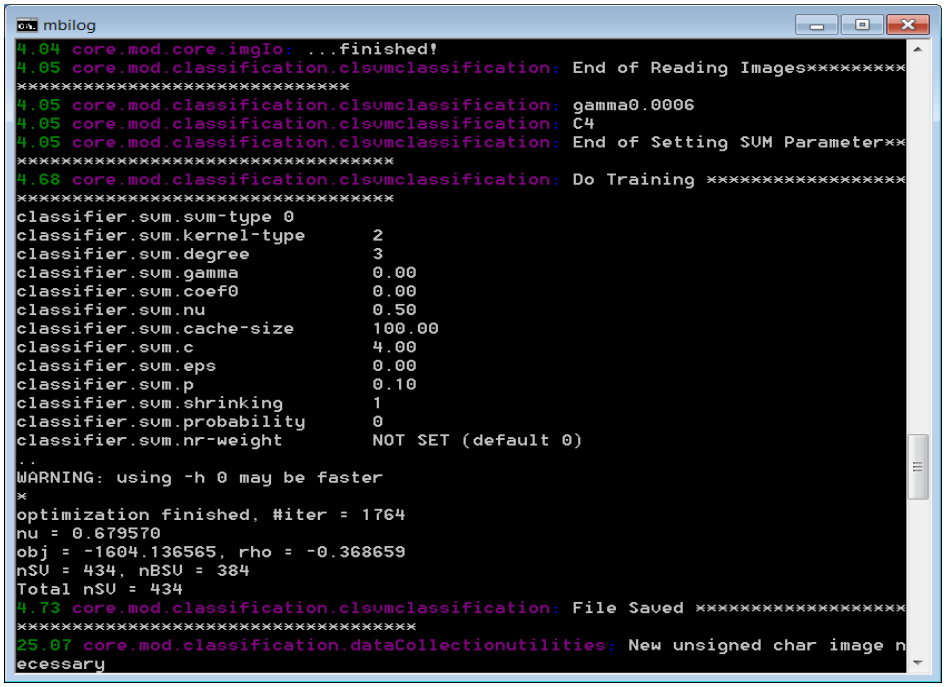


Abbildung 25: Durchführung des Trainings der entwickelten SVM

c) Für die Verwendung der Wrapperklasse „LibSVMClassifier.cpp“ wurde die Headerdatei der Wrapperklasse in die selbst implementierte SVM-Anwendung inkludiert. Damit ist es möglich, dass die integrierte SVM Methoden der Wrapperklasse für den Klassifikationsprozess nutzen kann.

d) Nach dem Training nutzt die implementierte SVM die Predict-Methode der Wrapperklasse, um so (nach Kapitel 3.1 Schritt 4) die Lebertumore zu klassifieren (siehe Abbildung 26) und die prädizierten Ergebnisbilder zu persistieren.

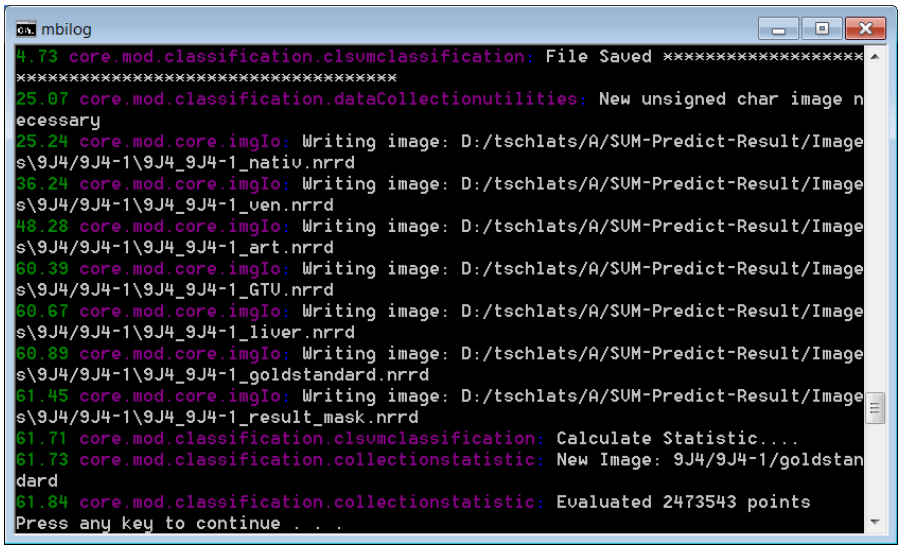


Abbildung 26: Durchführung der Prädiktion mit der entwickelten SVM

Die von der Wrapperklasse verwendete Methode Predict arbeitet nach folgendem Prinzip: `VectorY = predict(TestMatrixX)`

$$TestMatrixX = \begin{pmatrix} 122 & 34 & 99 \\ 24 & 87 & 42 \\ 4 & 203 & 1 \end{pmatrix}$$

Iteration:	Prädizierte Vektorelemente	Testmatrix (Matrix X)
1	$VectorY = \begin{pmatrix} 1 \\ ? \\ ? \end{pmatrix}$	$TestMatrixX = \begin{pmatrix} 122 & 34 & 99 \\ 24 & 87 & 42 \\ 4 & 203 & 1 \end{pmatrix}$
2	$VectorY = \begin{pmatrix} 1 \\ 2 \\ ? \end{pmatrix}$	$TestMatrixX = \begin{pmatrix} 122 & 34 & 99 \\ 24 & 87 & 42 \\ 4 & 203 & 1 \end{pmatrix}$
3	$VectorY = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$	$TestMatrixX = \begin{pmatrix} 122 & 34 & 99 \\ 24 & 87 & 42 \\ 4 & 203 & 1 \end{pmatrix}$

Abbildung 27: Mathematische Modellierung des Prädizierens der SVM

Gemäß Abbildung 27 entspricht VektorY und TestMatrixX sinngemäß dem Inhalt aus Kapitel 3.1 Formel (12). Im ersten Schritt prädiziert die SVM für die Voxelelemente der TestMatrixX in Zeile 1 gesundes Lebergewebe (Wert 1 im Prädiktionsvektor) und schreibt das Ergebnis in den VectorY. Der zweite Iterationsschritt der Prädiktion klassifiziert für den zweiten Zeilenvektor der TestMatrixX Tumorgewebe (Wert 2 im Prädiktionsvektor) und erweitert den VectorY entsprechend. Dieser Vorgang wird so lange wiederholt, bis der Zeilenrang der TestMatrixX, (also die maximale Anzahl an Zeilenvektoren) erreicht ist. Als Ergebnis der Prädiktion erhalten wir den vollständigen VectorY, dessen Inhalt äquivalent ist mit dem der prädizierten Tumorsegmentierung.

e) Zur Umsetzung von Softwareanforderung e) aus Kapitel 3.2 siehe Kapitel 4.
f) Die Umsetzung von Softwareanforderung f) aus Kapitel 3.2 findet sich in Kapitel 5.1.5.3. *Bemerkung:* Bei Verwendung von zu vielen gaußgeglätteten Bildern stürzt die entwickelte SVM ab. Dies liegt aber daran, dass nicht genug physikalischer Speicher auf dem Computer vorhanden ist, welcher die SVM Softwareapplikation nutzt.

4 Automatisierter Test

Dieses Kapitel umfasst den Bereich des Testens und Validierens der entwickelten LIBSVM durch das eigenentwickelte Programm „mitkLibSVMClassifierTestSuite“. Dabei wird zum einen getestet, ob Matrixobjekte identisch sind. Zum anderen wird getestet wie gut der Klassifikatior anhand einer implementierten Testmethode prädiziert, welche die cross-validation accuracy (CVA) berechnet. Zusammengefasst enthält die entwickelte Testklasse folgende Funktionen:

- Generiert aus einer CSV-Datei ein Matrizenpaarobjekt, welches dann dem Trainings- und Testdatensatz entspricht. Relevant für den Trainingsprozess und das Prädizieren.
- Einen CSV-Reader, welcher die eingelesene CSV-Datei in eine Matrixform konvertiert. Relevant für die Datenintegration.
- Einen Writer, der ein Matrixobjekt nach einer bestimmten Form in eine CSV-Datei konvertiert. Relevant für das mitgelieferte Validierungsprogramm der LIBSVM zur Parameteroptimierung.
- Eine Klassifikationsfunktion, welche das Training und die Prädiktion durchführt.
- Eine Testfunktion, welche den prädizierten Vektor des Testes mit dem tatsächlichen Prädiktionsvektor vergleicht und aufgrund des Ergebnisses eine Accuracy berechnet. Relevant für die Güte des Testes.
- Eine Testfunktion, die berechnet, ob sich der prädizierte Vektor in einem Toleranzintervall befindet. Relevant für die Beurteilung der Prädiktion.
- Eine Vergleichsfunktion, welche ermittelt, ob Matrixobjekte vom Inhalt her äquivalent sind.

4.1 Trainingsdaten und Testdaten

Für die Validierung der entwickelten SVM wurden zwei Datensätze verwendet, die jeweils in einen Trainings- und einen Testdatensatz aufgeteilt wurden. Ein Datensatz wurde von der LIBSVM library data (Homepage) heruntergeladen und beinhaltet Patientendaten mit einem Brustkrebskarzinom. Der andere Datensatz wurde mit Hilfe des Programmes Matlab selbst erzeugt. Für die Erzeugung des Matlabdatensatzes ist ein kleines Programm geschrieben worden, welches mittels einer mehrdimensionalen Gaußnormalverteilung randomisierte Daten produziert und anschließend gaußnormalverteilt. (siehe Abbildung 28)

```
A = [];  B = [];
for i = 1:350
a = rand();
   if(a < 0.5)
    xA = [normrnd(2,3),normrnd(2,1),normrnd(2,1)];
    A(i,:) = xA;
    hold on
    scatter3(normrnd(2,3),normrnd(2,1),normrnd(2,1));
    y1 = 1;
    B(i,:) = y1;
   end

   if(a > 0.5)
    xB = [normrnd(15,2),normrnd(15,1),normrnd(15,3)];
    A(i,:) = xB;
    scatter3(normrnd(15,2),normrnd(15,1),normrnd(15,3));
    y2 = 2;
    B(i,:)= y2;
   end
end
csvwrite('C:\Users\Alex\Desktop\Xmatrix.csv',A)
type 'C:\Users\Alex\Desktop\Xmatrix.csv'
csvwrite('C:\Users\Alex\Desktop\Ymatrix.csv',B)
type 'C:\Users\Alex\Desktop\Ymatrix.csv'
```

Abbildung 28: Matlabprogramm zur Erzeugung von normalverteilen Daten

Erzeugung von zwei gaußnormalverteilten Datensätzen A und B, welche nach Ausführung des Programmes in einem Streudiagramm geplottet und parallel in jeweils eine CSV-Datei ausgelesen werden.

Das Programm aus Abbildung 28 erzeugt 350 randomisierte Daten und teilt den Datensatz in einem Verhältnis von 50:50 in einen Datensatz A und einen Datensatz B ein. Die Datensätze A und B werden mit verschiedenen Varianzen und unterschiedlichen Erwartungswerten gaußnormalverteilt. Nach Ausführung des Programmes in Matlab werden die Datensätze (A und B) als Streudiagramm (Scatterplot) geplottet. (siehe Abbildung 29). Zudem werden die Datensätze jeweils in eine CSV-Datei geschrieben, damit mein entwickeltes Testprogramm diese Datensätze für die Validierung der entwickelten SVM nutzen kann.

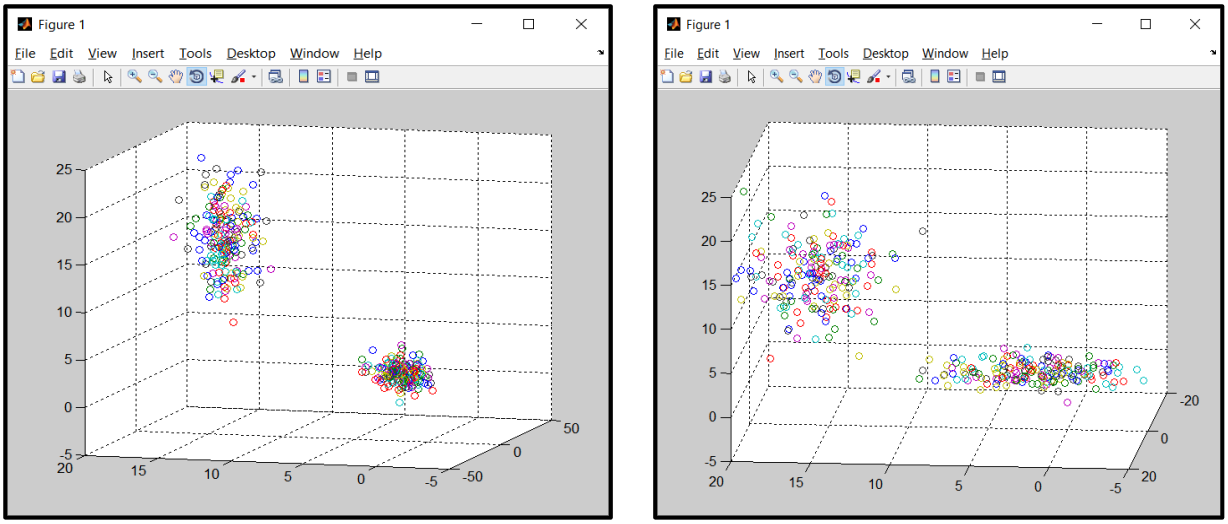


Abbildung 29: Scatterplot der gaußnormalverteilen Datensätze aus verschiedenen Perspektiven

Die linke Datenwolke entspricht dem normalverteilten Datensatz B mit einem Erwartungswertvektor von $\mu_{xyz} = (15{,}15{,}15)^T$ *und dem Varianzvektor:* $\sigma_{xyz} = (2{,}1{,}3)^T$. *Rechts zu sehen ist die Datenwolke des Datensatzes A mit:* $\mu_{xyz} = (2{,}2{,}2)^T$ *und* $\sigma_{xyz} = (3{,}1{,}1)^T$. *Dabei gibt der Erwartungswertvektor* μ_{xyz} *das Zentrum der jeweiligen Punktwolke an, der Varianzvektor* σ_{xyz} *die Streckung in x-, y- und z-Richtung.*

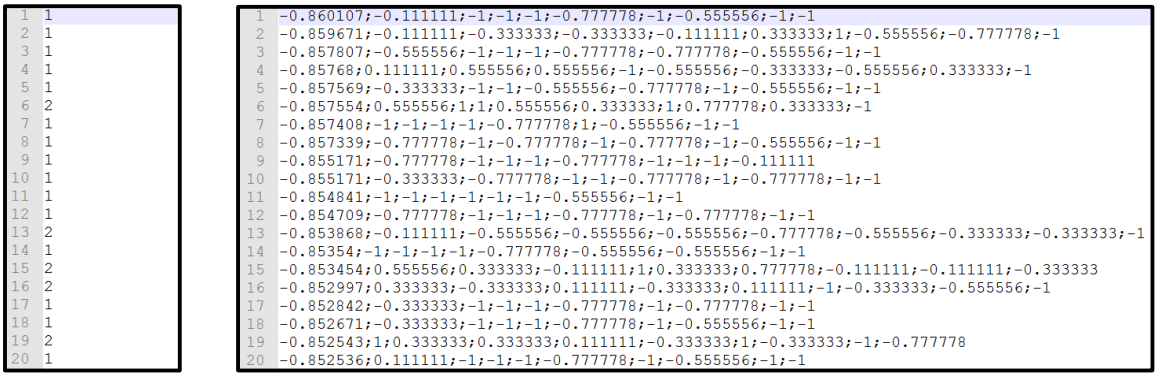
Erwartungsgemäß soll der entwickelte Test im Kapitel 4.3 zeigen, dass der Matlab-Datensatz aufgrund der Datenkonstellation und der geringeren Menge von Daten mit der SVM möglichst schnell und gut klassifiziert werden kann. Im Gegensatz dazu wird es noch einen weitereren Testfall geben, der aufgrund der nicht statistisch verteilten Daten und der größeren Datenmenge ein anderes Testergebis liefern soll.

Integration der bestehenden Datensätze in Trainings- und Testdaten

Der Datensatz des Patienten mit einem Brustkarzinom wurde in zwei CSV-Dateien ausgelagert. Eine CSV-Datei „LabelmatrixBreastcancer.csv“(Abbildung 30b links) enthält die erste Spalte des Datensatzes in Integer-Werten. Die andere CSV-Datei inkludiert die restlichen Spalten des Gesamtdatensatzes „FeaturematrixBreastcancer.csv“(Abbildung 30b rechts) mit Gleitkommata -Werten. Analog wurden auch die Matlab-Daten in CSV Dateien „LabelmatrixMatlab.csv“ und „FeaturematrixMatlab.csv“ konvertiert. (siehe Abbildung 30a)

$$Labelmatrix < int > = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 2 \end{pmatrix} \quad Featurematrix < double > = \begin{pmatrix} 2.5 & 6.1 & 10.0 \\ 3.2 & 5.0 & 12.3 \\ 5.1 & 1.3 & 22.4 \\ 0.3 & 3.9 & 18.9 \end{pmatrix}$$

Abbildung 30a: Inhalt der CSV-Dateien exemplarisch dargestellt



```
1
1
1
1
1
2
1
1
1
1
1
1
2
1
2
2
1
1
2
1
```

```
-0.860107;-0.111111;-1;-1;-1;-0.777778;-1;-0.555556;-1;-1
-0.859671;-0.111111;-0.333333;-0.333333;-0.111111;0.333333;1;-0.555556;-0.777778;-1
-0.857807;-0.555556;-1;-1;-1;-0.777778;-0.777778;-0.555556;-1;-1
-0.85768;0.111111;0.555556;0.555556;-1;-0.555556;-0.333333;-0.555556;0.333333;-1
-0.857569;-0.333333;-1;-1;-0.555556;-0.777778;-1;-0.555556;-1;-1
-0.857554;0.555556;1;1;0.555556;0.333333;1;0.777778;0.333333;-1
-0.857408;-1;-1;-1;-1;-0.777778;1;-0.555556;-1;-1
-0.857339;-0.777778;-1;-0.777778;-1;-0.777778;-1;-0.555556;-1;-1
-0.855171;-0.777778;-1;-1;-1;-0.777778;-1;-1;-1;-0.111111
-0.855171;-0.333333;-0.777778;-1;-1;-0.777778;-1;-0.777778;-1;-1
-0.854841;-1;-1;-1;-1;-1;-1;-0.555556;-1;-1
-0.854709;-0.777778;-1;-1;-1;-0.777778;-1;-0.777778;-1;-1
-0.853868;-0.111111;-0.555556;-0.555556;-0.555556;-0.777778;-0.555556;-0.333333;-0.333333;-1
-0.85354;-1;-1;-1;-1;-0.777778;-0.555556;-0.555556;-1;-1
-0.853454;0.555556;0.333333;-0.111111;1;0.333333;0.777778;-0.111111;-0.111111;-0.333333
-0.852997;0.333333;-0.333333;0.111111;-0.333333;0.111111;-1;-0.333333;-0.555556;-1
-0.852842;-0.333333;-1;-1;-1;-0.777778;-1;-0.777778;-1;-1
-0.852671;-0.333333;-1;-1;-1;-0.777778;-1;-0.555556;-1;-1
-0.852543;1;0.333333;0.333333;0.111111;-0.333333;1;-0.333333;-1;-0.777778
-0.852536;0.111111;-1;-1;-1;-0.777778;-1;-0.555556;-1;-1
```

Abbildung 30b: Genauer Inhalt des Brustkrebsdatensatzes in CSV-Format

In Abbildung 30a illustriert die Labelmatrix die vereinfachte Darstellung von Abbildung 30b links. Der Inhalt in der CSV-Datei in Abbildung 30b rechts kann als Matrix interpretiert werden. (siehe Abbildung 30a Featurematrix) Die Bedeutung der CSV-Dateien Labelmatrix und Featurematrix ist äquivalent zu der Formel (11) in Kapitel 3.1.

Damit im nächsten Schritt die CSV-Dateien in Trainings- und Testdaten konvertiert werden können, wird eine implementierte Methode genutzt, die als formale Parameter eine CSV-Datei, ein Trennzeichen (engl. delimiter), eine Reichweite (engl. range) und einen booleschen Operator verlangt. Die Methode liefert ein Matrixpaarobjekt zurück. In diesem Fall initialisieren wir ein Matrixpaarobjekt und geben diesem Pfad (siehe unten) das Trennzeichen **„;“,** eine Reichweite von 50% und den booleschen Wert true für eine Featurematrix (siehe Abbildung 31a) mit. Für die Generierung einer Labelmatrix wird boolesche Wert einfach auf false gesetzt. (siehe Abbildung 31b).

Codeauszug 3: Konvertierung einer CSV in ein Matrixpair „„mitkLibSVMClassifier-TestSuite“.

```
/* Declarating an featurematrixdataset, the first matrix
of the matrixpair is the trainingmatrix and the second one is the test-
matrix.*/
std::pair<MatrixDoubleType, MatrixDoubleType> matrixDouble;
matrixDouble = convertCSVToMa-
trix<double>("D:/tschlats/A/Classification/FeaturematrixBreastcancer.csv",
';', 0.5, true);
m_TrainingMatrixX = matrixDouble.first;
m_TestXPredict = matrixDouble.second;
```

Das Ergebnis der Methode (aus Codeauszug 3) sieht beispielsweise wie folgt aus:

$$a)\ Featurematrix.csv = \begin{pmatrix} 2.5 & 6.1 & 10.0 \\ 3.2 & 5.0 & 12.3 \\ 5.1 & 1.3 & 22.4 \\ 0.3 & 3.9 & 18.9 \end{pmatrix} \rightarrow \frac{\begin{pmatrix} \mathbf{2.5} & \mathbf{6.1} & \mathbf{10.0} \\ \mathbf{3.2} & \mathbf{5.0} & \mathbf{12.3} \end{pmatrix}}{\begin{pmatrix} \mathbf{5.1} & \mathbf{1.3} & \mathbf{22.4} \\ \mathbf{0.3} & \mathbf{3.9} & \mathbf{18.9} \end{pmatrix}} \quad range = 50\%$$

$$b)\ Labelmatrix.csv = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 1 \end{pmatrix} \rightarrow \frac{\begin{pmatrix} \mathbf{1} \\ \mathbf{2} \end{pmatrix}}{\begin{pmatrix} \mathbf{2} \\ \mathbf{1} \end{pmatrix}} \quad range = 50\%$$

Abbildung 31: Konvertierung einer CSV Datei in ein Matrixpaar

Die rote Matrix entspricht dem Trainingsdatensatz, die blaue Matrix dem Testdatensatz der eingegebenen CSV-Datei. Die Range entspricht dem Prozentsatz mit dem die Matrix in ein Matrixpaar (Training/Test) aufgeteilt wird, orientiert an der Gesamtzahl der Zeilen der Matrix.

Analog gilt die Konvertierung in (Codeauszug 3) auch für die Labelmatrix. Bei Instanzierung einer Labelmatrix wird der Templateparameter (aus Codeauszug 3) von double auf int gesetzt. Die Methode convertCSVToMatrix(..) wird jetzt für die CSV-Dateien des Patienten mit Brustkrebs und für die in Matlab generierten CSV-Dateien verwendet, um die Daten in Trainings- und Testdatensätze (in Form von Matrizenpaaren) zu konvertieren. (siehe Codeauszug 3). Mit der Operation matrix.first wird auf die Trainingsmatrix (siehe Abbildung 31 rote Matrix), mit matrix.second auf die Testmatrix (siehe Abbildung 31 blaue Matrix) zugegriffen.

4.2 Evaluation der optimalen Konfiguration der SVM Parameter

Zur Bestimmung der optimalen Parameterkonfiguration im Testprogramm wird das mitgelieferte Kommandozeilenprogramm der LIBSVM verwendet. Dafür ist es notwendig, dass ein valider Trainingsdatensatz für das Optimierungsprogramm bereitgestellt wird. Für die Parameteroptimierung des Brustkrebsdatensatzes wird folgender Trainingsdatensatz verwendet: (siehe Abbildung 32)

```
1;-0.860107;-0.111111;-1;-1;-1;-0.777778;-1;-0.555556;-1;-1
1;-0.859671;-0.111111;-0.333333;-0.333333;-0.111111;0.333333;0;-0.555556;-0.777778;-1
1;-0.857807;-0.555556;-1;-1;-1;-0.777778;-0.777778;-0.555556;-1;-1
1;-0.85768;0.111111;0.555556;0.555556;-1;-0.555556;-0.333333;-0.555556;0.333333;-1
1;-0.857569;-0.333333;-1;-1;-0.555556;-0.777778;-1;-0.555556;-1;-1
2;-0.857552;0.555556;0;0;0.555556;0.333333;0;0.777778;0.333333;-1
1;-0.857208;-1;-1;-1;-1;-0.777778;0;-0.555556;-1;-1
1;-0.857339;-0.777778;-1;-0.777778;-1;-0.777778;-1;-0.555556;-1;-1
1;-0.855171;-0.777778;-1;-1;-1;-0.777778;-1;-1;-1;-0.111111
1;-0.855171;-0.333333;-0.777778;-1;-1;-0.777778;-1;-0.777778;-1;-1
1;-0.852821;-1;-1;-1;-1;-1;-1;-0.555556;-1;-1
1;-0.852709;-0.777778;-1;-1;-1;-0.777778;-1;-0.777778;-1;-1
2;-0.853868;-0.111111;-0.555556;-0.555556;-0.555556;-0.777778;-0.555556;-0.333333;-0.333333;-1
1;-0.85352;-1;-1;-1;-1;-0.777778;-0.555556;-0.555556;-1;-1
2;-0.853252;0.555556;0.333333;-0.111111;0;0.333333;0.777778;-0.111111;-0.111111;-0.333333
2;-0.851997;0.333333;-0.333333;0.111111;-0.333333;0.111111;-1;-0.333333;-0.555556;-1
1;-0.851821;-0.333333;-1;-1;-1;-0.777778;-1;-0.777778;-1;-1
1;-0.851671;-0.333333;-1;-1;-1;-0.777778;-1;-0.555556;-1;-1
2;-0.851523;0;0.333333;0.333333;0.111111;-0.333333;0;-0.333333;-1;-0.777778
1;-0.851536;0.111111;-1;-1;-1;-0.777778;-1;-0.555556;-1;-1
```

Abbildung 32: Trainingsdatensatz (Brustkrebspatient) im validen CSV-Format

Bei genauerer Betrachtung von Abbildung 32 fällt auf, dass diese eine Kombination aus den Teilabbildungen von Abbildung 30b ist. Startet man das Validierungsprogramm der SVM, variiert die SVM-Parameter und setzt die 10-fache Kreuzvalidierung ein, kommt man bei folgender Parametereingabe zu einer optimalen Accuracy in Höhe von 97,6574**%.** (siehe Abbildung 33)

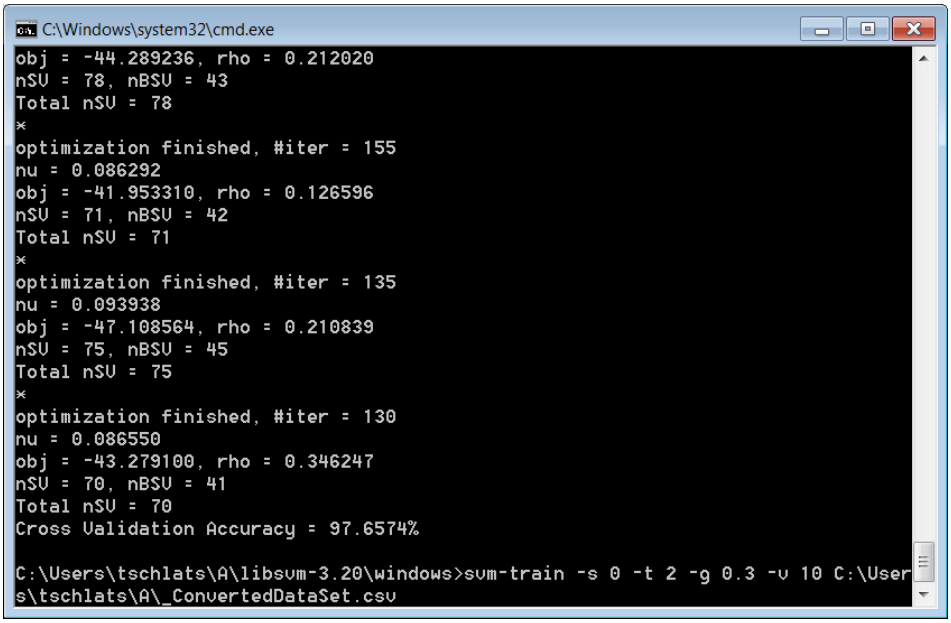


Abbildung 33: Optimale Parameterkonfiguration des Brustkrebsdatensatzes

Die Parameteroptimierung für den Matlabdatensatz verläuft analog zu den oben beschriebenen Fall.

4.3. Das Training und die Prädiktion der Testklasse

Die Testklasse beinhaltet zwei Testszenarien: TEST(TrainSVMClassifier_MatlabDataSet_shouldReturnTrue), ein Testszenario für den Matlabdatensatz, und ein anderes für den Brustkrebsdatensatz TEST(TrainSVMClassifier_BreastCancerDataSet_shouldRetunTrue). Beide Testszenarien enthalten jeweils einen spezifisch geschriebenen Test, zur Validierung der entwickelten LIBSVM.
Während der Ausführung des Testes wird die Trainmethode der Wrapperklasse „mitk-LibSVMClassifier.cpp“ aufgerufen, welche dann als aktuelle Parameter die Trainingsmatrizen (vgl. Kapitel 4.1 Abbildung 31 rote Matrizen) erhält und somit eine Modeldatei trainiert. Direkt danach wird die Methode predict() der Wrapperklasse verwendet, um einen Prädiktionsvektor zu klassifizieren. Der Test berechnet jetzt wie genau der Prädiktionsvektor mit dem erwarteten Klassifikationsergebnis übereinstimmt. (siehe Abbildung 35 predicted accuracy). Durch die Verwendung von Matrixpaarobjekten, die wie im Kapitel 4.1 in Trainings- und Testbestandteile aufgeteilt wurde, hat man das erwartete Klassifikationsergebnis, welches die Klassifikation erreichen soll, schon vor dem Prädizieren. Dieses Ergebnis ist die Testmatrix des Matrizenpaarobjektes der Labelmatrix des jeweiligen Datensatzes (siehe Kapitel 4.1 Abbildung 31 blaue Matrizen). Anschließend wird überprüft, ob sich der prädizierte Vektor in einem zu vor berechneten Toleranzintervall befindet. Ist dies der Fall, ist der Test bestanden [PASSED], falls nicht, erscheint die Meldung [FAIL]. (Abbildung 35)
Wird die Testklasse mit insgesamt zwei Tests ausgeführt, stellt man fest, dass für den Matlabdatensatz eine Accuracy von 88% und für die des Brustkrebsdatensatzes eine Accuracy von 76,6% berechnet wurde. (siehe Abbildung 35)
Schauen wir uns den Matlabdatensatz an, besteht dieser aus insgesamt 350 Daten, davon sind 172 Daten gesundem Lebergewebe und 178 Daten Lebertumorgewebe zugeordnet. Somit liegt die relative Häufigkeit von gesundem Lebergewebe bei 49,14% und die von krankem Tumorgewebe bei 50,86%. Das bedeuet, dass im gesamten Matlabdatensatz das Verhältnis von gesundem Lebergewebe zu tumorhaftigem Gewebe gleich eins zu eins ist. (1 : 1) Im Brustkrebsdatensatz sieht das Verhältnis von Tumor zu gesundem Gewebe wie folgt aus: Insgesamt beinhaltet der Datensatz 683 Daten. Davon sind 444 Daten (65,01%) gesundem Lebergewebe und 239 Daten (34,99%) Tumorgewebe zugeordnet. Hier kann man sagen, dass das Verhältnis von gesundem Gewebe zum Tumorgewebe bei zwei zu eins liegt (2 : 1). Die ermittelten Erkenntnisse sind relevant, um im nächsten Schritt beurteilen zu können, wie gut die SVM in diesen Testfällen tatsächlich klassifiziert hat. Die berechnete Accuracy des Matlabdatensatzes von 88% ist ein sehr gutes Ergebnis, da das Verhältnis von Nichttumor zu Tumor bei (1:1) liegt. Das bedeutet, selbst wenn 50% Tumorgewebe in der berechneten Accuracy enthalten sein sollten, so sind immer noch 38% richtig klassifiziertes gesundes Lebergewebe darunter. Umgekehrt gilt dies genauso. Aus diesem Grunde besteht der Toleranzintervalltest. Die Grenzen des Intervalles wurden so berechnet, dass die jeweiligen Verhältnisse von Nichttumor zu Tumor darin berücksichtigt sind. Auch die berechnete Accuracy des Brustkrebsdatensatzes in Höhe von 76,6% ist unter Beachtung der oben genannten Relationen ein gutes Ergebnis. Das heißt nehmen wir an, dass 34.99% richtig klassifiziertes Tumorvolumen ist, so hätten wir immer noch 41,61% korrekt klassifiziertes Lebergewebe (bei einer Accuracy von 76,6%). Deswegen ist die im Test inkludierte Intervalluntergrenze für Brustkrebs kleiner als die der Matlabdaten.

Die Testergebnisse sind in beiden Fällen sehr positiv verlaufen. (Abbildung 35) Beide verwendeten Datensätze enthielten ein unterschiedliches Verhältnis von Tumor zu nicht Tumor. Daraus folgt, dass die entwickelte SVM in hohem Maße richtig klassifiziert. Das Testergebnis ist im Fall des Matlabdatensatzes um 11,4% besser als das des Brustkrebsdatensatzes. Grund dafür ist die statistische Verteilung der Daten. Im Kapitel 4.2 wurde bei der Parameteroptimierung des Brustkrebsdatensatzes eine Accuracy von 97,6574**%** berechnet, aber der Test prädizierte 76,61% richtig. Dies hat folgenden Grund: Im Fall der Parameteroptimierung in Kapitel 4.2 wurde die Technik der Kreuzvalidierung verwendet, um so die optimale Parameterkonfiguration zu erhalten. (siehe Abbildung 34):

K = 5	Datenmenge				
Iteration 1	Test	Training	Training	Training	Training
Iteration 2	Training	Test	Training	Training	Training
Iteration 3	Training	Training	Test	Training	Training
Iteration 4	Training	Training	Training	Test	Training
Iteration 5	Training	Training	Training	Training	Test

Abbildung 34: 5-fache Kreuzvalidierung

Bei einer Kreuzvalidierung werden die gesamten Daten zufällig in k Teilmengen unterteilt. Am Beispiel von Abbildung 34 wird die gesamte Datenmenge in fünf Teilmengen aufgeteilt. Im ersten Iterationsschritt trainiert die SVM mit der zweiten, dritten und vierten Teilmenge und prädiziert (testet) mit Hilfe der ersten Teilmenge (siehe Abbildung 34 erste Zeile). Dieser Vorgang wird dann noch vier weitere Male wiederholt, wobei der Testdatensatz pro Iteration immer eine Teilmenge weiterwandert. (siehe Abbildung 34) Aufgrund der Verwendung des Kreuzvalidierungsverfahrens im mitgelieferten Validierungsprogramm der SVM wurde der Brustkrebsdatensatz in zehn Teile aufgeteilt und validiert. In der Testklasse wird der Gesamtdatensatz bei Durchführung des Testes in jeweils zwei Teile aufgeteilt. Die größere Aufteilungsrate und Datenmenge des Brustkrebsdatensatzes durch die Kreuzvalidierung führte letztendlich zu einer höheren Accuracy als die berechnete.

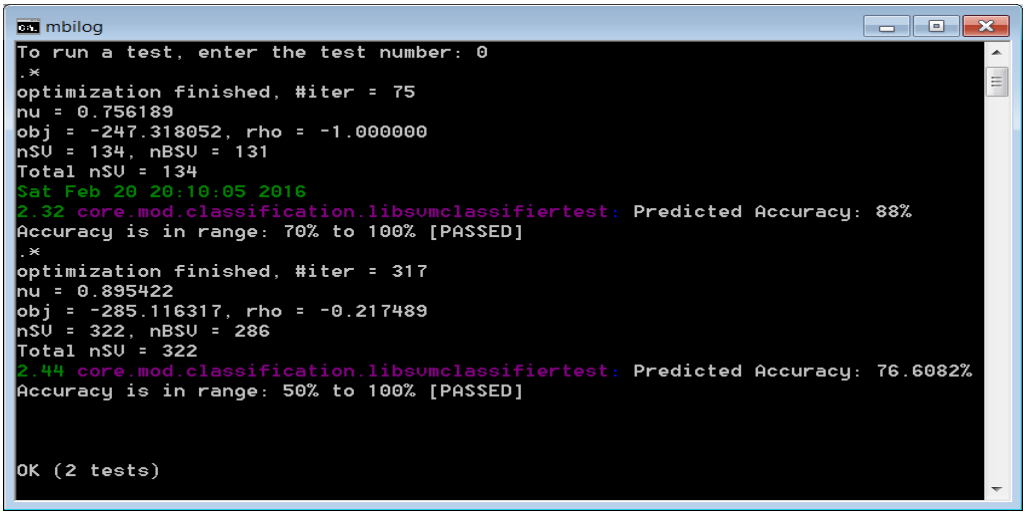


Abbildung 35: Durchführung der Testklasse „mitkLibSVMClassifierTestSuite**“**

5 Real-Life-Test

Im folgenden Kapitel geht es darum, mit der entwickelten SVM eine vollautomatische Lebertumorsegmentierung durchzuführen.

5.1 Datenintegration

Bevor eine automatische Lebertumorsegmentierung durchgeführt werden kann, benötigen wir Patientendaten. Die entsprechenden Daten wurden in der XNAT-Datenbank des Deutschen Krebsforschungszentrums in Heidelberg bereitgestellt. Dabei handelt es sich um CT-Bilddatensätze von insgesamt drei Patienten mit Lebertumoren. Der erste Schritt ist es, diese bildbasierten Datensätze (DICOM-Daten) in eine Verzeichnisstruktur einzubinden. Hierfür wird für jeden Patienten ein Ordnerverzeichnis angelegt. Nach der Anlage der Daten ist der nächste Schritt, diese in die bestehende LIBSVM Applikation zu integrieren. Zur Datenintegration werden eine XML-Datei und eine Textdatei benötigt. Die XML-Datei wird dann nach dem gleichen Prinzip aufgebaut, wie dies in (Kapitel 3.1) im Schritt „Vorbereitung der Trainingsdaten für das Training“ erläutert wurde. (siehe Kapitel A.2 XML-Datei) Die Textdatei fungiert dabei als Übergabeparameter für die entwickelte LIBSVM Anwendung, indem die Bildmodalitäten, die Trainings- und die Testmaske einfach dieser Textdatei übergeben werden. Jedes Bild, das der Anwender für die Testsegmentierung haben möchte, wird zuerst in die XML-Datei eingebettet. Für diesen Real-Life-Test wurden gemäß (Kapitel A.2 XML-Datei) insgesamt 15 Bilder verwendet, d.h. 5 Bilder pro Patient (art, ven, nativ, liver und GTV). Abbildung 36 illustriert eine Bildserie, in der jede transversale Bildaufnahme dem Krankheitsbild eines Patienten entspricht.

Abbildung 36: Datenkollektiv für den Real-Life-Test

Zu sehen sind 3 Patienten: Von links nach rechts Patient A, Patient B und Patient C. Bei jedem Patienten sind eine Nativaufnahme (Grauwertbild transparent im Hintergrund) sowie die morphologische Struktur der Leber mit Lebertumor(Goldstandard) (Farbbild mit hellem Tumorbereich) dargestellt.

5.1.1 Definition der Trainings- und Testmasken

Im nächsten Schritt werden die Trainings- und Testmasken definiert, welche die SVM für den Klassifikationsprozess (vgl. Kapitel 3.1) benötigt. Insgesamt sind 3 Patienten im Datenkollektiv vorhanden. (siehe Abbildung 36) Dies bedeutet, dass immer die Daten von zwei Patienten für den Trainingsdatensatz bereitgestellt werden und die eines Patients für den Testdatensatz. Zur Durchführung der automatischen Lebersegmentierung wird jeweils für Patient A und Patient B eine Trainingsmaske und für Patient C eine Testmaske definiert. Dafür wird wie folgt vorgegangen:

Definierung der Trainingsmasken

Bei Definierung einer Trainingsmaske werden zuerst alle Bildmodalitäten (arteriell, venös und nativ) des jeweiligen Patienten übereinandergelegt. Anschließend wird der Grad der Transparenz des jeweiligen Bildes so angepasst, dass man eine eindeutige Schnittmenge der jeweiligen Tumorvolumina (dunkle Grauwerte innerhalb der Leber) (Abbildung 37) erkennen kann. Ist man sich sicher, dass der Tumorbereich eindeutig zu sehen ist, segmentiert man den Tumor möglichst präzise, um keine Grauwerte in die Trainingsmaske zu nehmen, die nicht im Tumorbereich sind. Die Tumorsegmentierung wird nun als Bild abgespeichert. (z.B. Livercancer.nrrd) Der gleiche Vorgang der Segmentierung wird wiederholt, aber diesmal wird gesundes Lebergewebe (helle Grauwerte) segmentiert. Auch dieses Ergebnis wird als Bild abgespeichert. (z.B. Liversegmentation.nrrd)

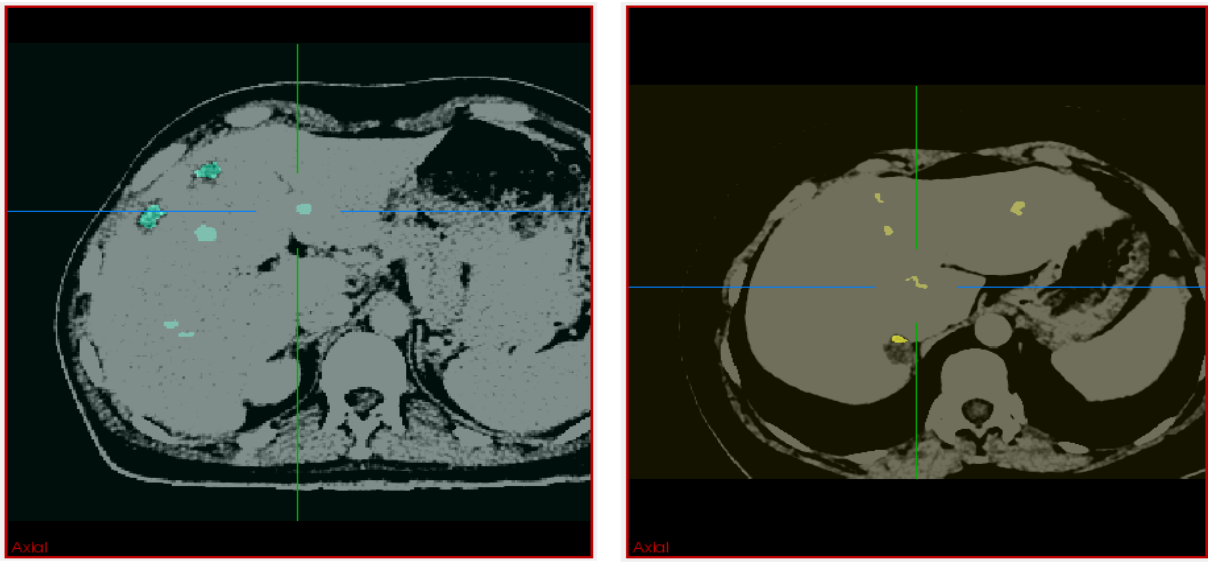


Abbildung 37: Segmentierung der Trainingsmasken mit MITK

Links ist die Trainingsmaske für Patient A abgebildet (siehe blau segmentierte Punkte), rechts ist die Trainingsmaske für Patient B zu sehen (orange segmentierte Punkte). In CT-Datensätzen sind Tumorvolumina dunkelgrau bis schwarz in der jeweiligen morphologischen Struktur dargestellt.

Wie in Kapitel 3.1 erwähnt benötigt unsere entwickelte LIBSVM ein Labelbild, da wir mindestens zwischen zwei Klassen, nämlich Klasse „Tumor“ und Klasse „kein Tumor“ differenzieren möchten. Bisher sind nur zwei einzelne Segmentierungsbilder mit einem Label vorhanden. Um dieses Problem zu lösen, müssen die segmentierten Bilder binär addiert werden. Es gibt zwei Möglichkeiten, Bilder binär zu addieren. Die einfachere Möglichkeit besteht darin das erste Bild zweimal im Datamanager von MITK hineinzuladen und diese Bilder binär zu addieren. Das Ergebnis ist ein Bild mit dem Labelwert 2 (hier für Tumorvolumen). Nun wird das Ergebnisbild mit dem Labelwert 2 und das zweite Bild mit dem Labelwert 1 in den Datamanager geladen und mithilfe des Plugins binär addiert. Als Ergebnis erhält man ein binäres Labelbild mit den Labelwerten 1 und 2. (siehe Abbildung 38)

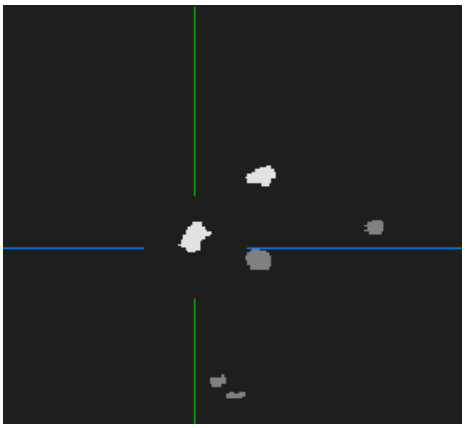


Abbildung 38: Trainingsmaske als Labelbild

Weiß dargestellt ist der Teil des Binärbildes mit dem Labelwert 2 für Tumorgewebe und grau definiert ist der Labelwert 1 für gesundes Lebergewebe.

Die zweite Möglichkeit besteht darin, die Bilder binär mithilfe der selber implementierten Anwendung „MitkCLAddImages“ zu addieren. Diesem Programm wird als Eingabeparameter der Pfad zu zwei einzelnen segmentierten Bildern übergeben. Anschließend besteht die Möglichkeit, mithilfe eines Samplefilters so genannte Subsamples zu generieren, also zufällige Stichproben von Grauwerten aus dem bestehenden Bild zu nehmen und beide Inputimages zu addieren. (siehe Codeauszug 4)

Codeauszug 4 Binäres Addieren von zwei Bildern „MitkCLAddImages“

```
itk::AddImageFilter<itk::Image<int, 3>, itk::Image<int, 3>, itk::Image<int,
3> >::Pointer filter = itk::AddImageFilter<itk::Image<int, 3>,
itk::Image<int, 3>, itk::Image<int, 3> >::New();

filter->SetInput1(itk_image_liver1);
filter->SetInput2(itk_image_gtv1);
filter->Update();
mitk::CastToMitkImage(filter->GetOutput(), Sample1outputIM);
```

Hierbei entspricht `itk_image_liver1` dem Lebersegmentierungsbild und `itk_image_gtv1` ist die Tumorsegmentierung der Leber. Mit dem Methodenaufruf `filter->Update()` werden beide Bilder addiert.

Definierung der Testmaske:

Als Testmaske wurde für den Real-Life-Test der Bereich der gesamten Leber gewählt. Die Segmentierung der Leber des Patienten C musste in diesem Fall nicht manuell durchgeführt werden, da diese bereits aus der Patientendatenbank zur Verfügung gestellt wurde. (siehe Abbildung 39)

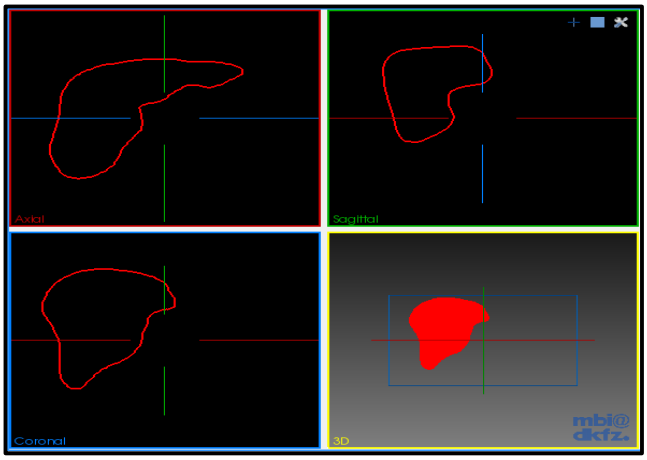


Abbildung 39: Testmaske in MITK

Rot dargestellt ist eine komplette Segmentierung der Leber des Patienten C. Diese wird der SVM als Testregion gegeben, da Tumore auf der kompletten Leber klassifiziert werden sollen. Es ist auch möglich, die Lebersegmentierung in MITK in den transversalen, sagittalen und frontalen Rendering-Windows farblich komplett ausgefüllt anzuzeigen.

5.1.2 Auswahl geeigneter Features

Um ein gutes Klassifikationsergebnis mit der SVM zu erhalten, ist es essentiell dem Klassifikator passende Merkmale zu übergeben. Zum einen werden der SVM standardmäßig die Grundmodalitäten übergeben: eine arterielle, eine venöse und eine native Bildaufnahme des Patienten. Dies ist notwendig, da die SVM aufgrund dieser Grauwertbilder die Leber mit den Tumoren in der richtigen Region des Grauwertbildes klassifizieren kann. Zum anderen werden der SVM die im Kapitel 5.1.1 definierten Trainings- und Testmasken übergeben. Die Trainings- und Testmasken (also die Labelbilder) werden dann Bestandteil des Trainings- und Testdatensatzes, welche die SVM für den Klassifikationsprozess benötigt (siehe Kapitel 3.1). Zusätzlich zu den bisher ausgewählten Merkmalen wurden für den Real-Life-Test noch Texturen (siehe Kapitel 2.3.2) in Form von gaußgeglätteten Bildern verwendet. Die Notwendigkeit der Verwendung von Texturen kann in Kapitel 2.3.2 nachgelesen werden. Die Glättung der Bilder erfolgte mit der bereitgestellten Anwendung „CLVoxelFeatures“ in MITK.

Grundsätzlich bietet das Kommandozeilenprogramm „CLVoxelFeatures“ folgende Möglichkeiten der Glättung von Bildern: modifiziert nach [HH 09]

Filter	Formel
Gaussian (G)	$GN(x,y) = f(x,y) \otimes G(x,y); \quad G(x,y) = \frac{1}{2\pi\sigma^2} \cdot e^{\frac{x^2+y^2}{2\sigma^2}}$
Difference of Gaussian (DoG)	$D(x,y) = f(x,y) \otimes \frac{1}{2\pi\sigma^2} \cdot e^{\frac{x^2+y^2}{2\sigma^2}} - f(x,y) \otimes \frac{1}{2\pi K^2\sigma^2} \cdot e^{\frac{x^2+y^2}{2K^2\sigma^2}}$
Laplacian of Gaussian (LoG)	$L(x,y) = \Delta G(x,y) = \nabla^2 G(x,y) = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2}$
Hessian of Gaussian (HoG)	$H(x,y) = \begin{pmatrix} \frac{\partial^2 GN(x,y)}{\partial x^2} & \frac{\partial^2 GN(x,y)}{\partial x \partial y} \\ \frac{\partial^2 GN(x,y)}{\partial y \partial x} & \frac{\partial^2 GN(x,y)}{\partial y^2} \end{pmatrix} \det\big(H(x,y)\big) = GN_{xx} \cdot GN_{yy} - GN_{xy}^2$

Damit ein Bild geglättet werden kann, benötigt die Anwendung „CLVoxelFeatures“ den Pfad des zu glättenden Bildes, die Wahl des Glättungsfilters, die Varianzen für die Glättung und den Pfad, in dem das geglättete Bild gespeichert werden soll. Diese Parameter werden der Anwendung als Programmparameter in folgender Form übergeben:

```
-i User/tschlats/A/9J4-art.nrrd –dog 1.5 –o User/tschlats/A/9J4art-dog.nrrd
```

Nach Angabe der Kommandozeilenparameter und Ausführung des Programmes „CLVoxelFeatures“ wird das eingegebene Bild Difference of Gaussian geglättet (siehe Abbildung 40)

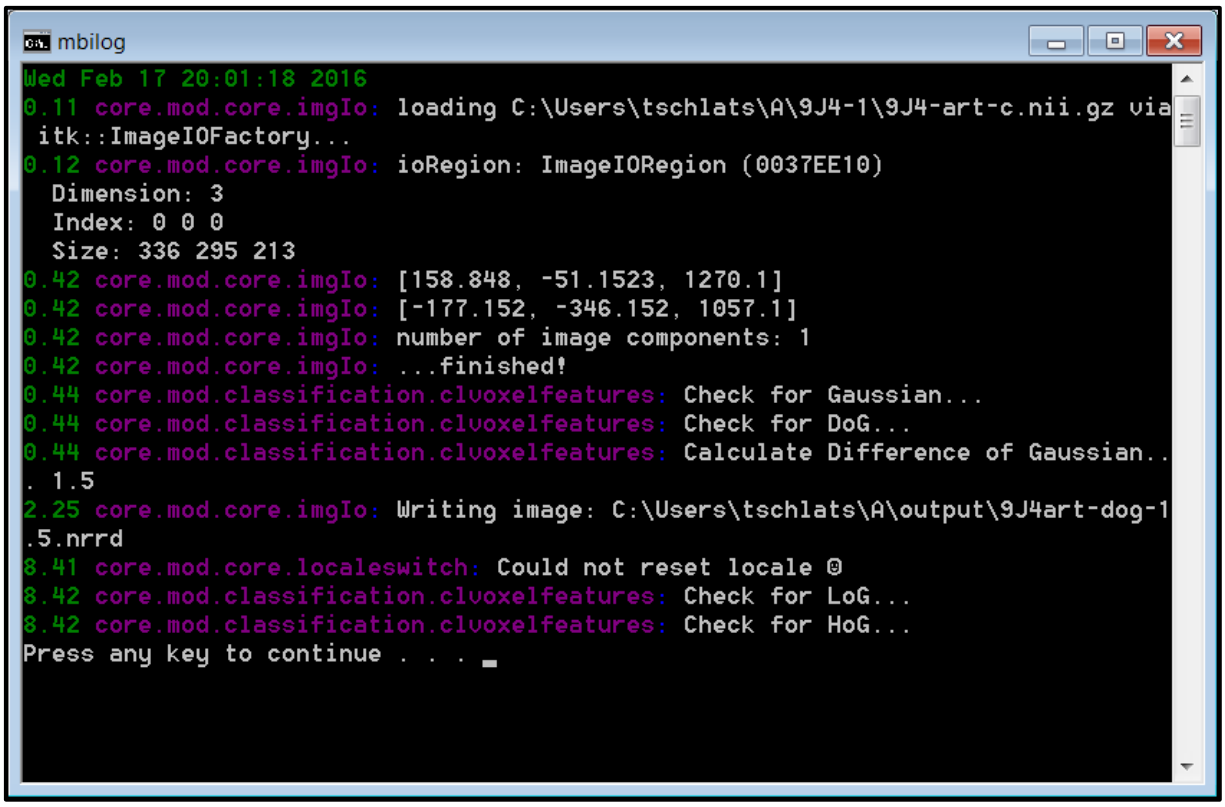


Abbildung 40: Kommandozeilenprogramm „CLVoxelfeatures“

Die Applikation zeigt unter loading den Pfad des geladenen Bildes, gefolgt von der Darstellung des Bildes in Matrixform. Angezeigt wird auch, welcher Filter für die Glättung verwendet und unter welchem Pfad das geglättete Bild gespeichert wurde.

Es ist auch möglich, durch Angabe von mehreren Varianzen mehrere geglättete Bilder gleichzeitig zu erstellen. Durch Verwendung von gaußgeglätteten Bildern kann das Klassifikationsergebnis marginal verbessert werden, da beispielsweise eine Glättung die Voxel besser interpoliert.

5.1.3 Parameteroptimierung

Bevor es mit dem Training des Real-Life-Tests losgehen kann, müssen zuvor die Parameter der SVM initialisiert und optimiert werden. Durch die Parameteroptimierung wird nach Durchführung des Trainings eine optimierte Modeldatei erstellt, welche zu einem guten Klassifikationsergebnis führt. Zur Parameteroptimierung wird das mitgelieferte Programm der LIBSVM genutzt. Dieses Kommandozeilenprogramm unterstützt zwei mögliche Funktionen:

- svm-train.exe (zum Trainieren)
- svm-predict.exe (für das Prädizieren)

Bei Ausführung des Befehles svm-train.exe gelangen wir in folgendes Menü: (siehe Abbildung 41)

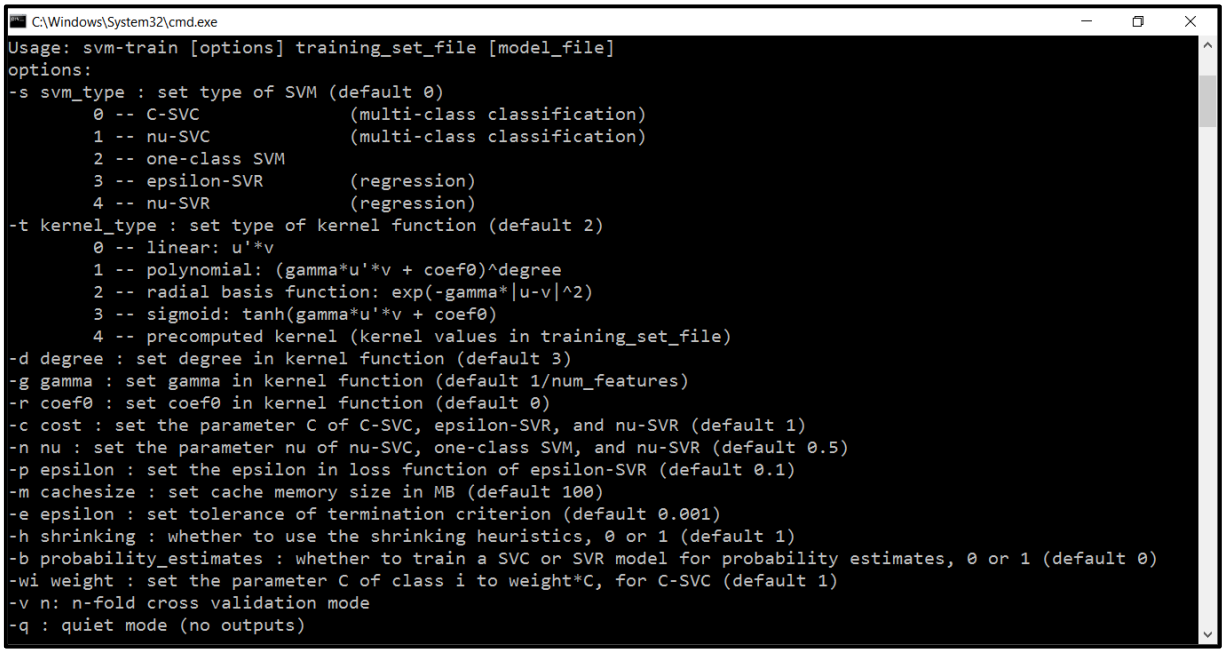


Abbildung 41: Kommandozeilenprogramm svm-train

Aufgelistet sind die Eingabeparameter der SVM, welche benötigt werden, um eine Modeldatei zu erstellen.

Zusätzlich kann bei Eingabe der SVM-Parameter das Kreuzvalidierungsverfahren aktiviert werden, welches eine Accuracy liefert (siehe Kapitel 2.4.1) Das Kreuzvalidierungsverfahren (Kapitel 4.3) hilft uns zu verstehen, wie sich ein Klassifikator verhält. Eine Kennzahl des Ergebnisses des Kreuzvalidierungsverfahrens ist die Accuracy, welche besagt, wie gut der Klassifikator, also die entwickelte LIBSVM, klassifiziert. Der nächste Schritt besteht darin, die SVM Parameter sowie die benötigte Trainingsdatenmatrix dem Kommandozeilenprogramm im folgendem Format zu übergeben:

$$Beispiel: -s\ 0 -t\ 2 -g\ 10^{-6} -c\ 4 -v\ 10 \quad C{:}\backslash Users\backslash Traindata\backslash train.txt$$

Zuvor sollte man sich überlegen, welche sinnvollen Parameterkonfigurationen in Frage kommen würden. Bei der Durchführung des Real-Life-Tests wurden die „multiclass-SVM“ (Parameter s 0) und der RBF-Kernel mit der besten Klassifikationsperformance (Parameter t 2) genutzt. Wird der RBF-Kernel und die multiclass-SVM verwendet, müssen auch die Parameter Gamma (Parameter g) und Kosten (Parameter c) dementsprechend gesetzt werden. Die k-fache Kreuzvalidierung wird mithilfe des Parameters v k gesetzt. Das heißt, wenn der Parameter v 10 gesetzt wird, so teilt sich der Gesamtdatensatz der Trainingsmatrix in zehn Teile. (siehe Kapitel 4.2) Vor Ausführung des Kommandozeilenprogramms zur Validierung der LIBSVM wird nun die implementierte LIBSVM „MitkCLSVMClassification“ einmal initial ausgeführt, um die Bilddaten in eine Textdatei zu konvertieren.

Codeauszug: 5 Konvertierung der Trainingsmatrix in eine Textdatei “MitkCLSVMClassification”. Die Konvertierung der Testmatrix erfolgt analog.

```
bool isTrainMatrix = true;

if (isTrainMatrix == true){
      MITK_INFO << "Save Trainmatrices into csv files";
      std::ofstream file("D:/tschlats/A/trainMatrix.txt");
      if (file.is_open())
                  {
            Eigen::MatrixXd trainDx = trainDataX;
            Eigen::MatrixXi trainDy = trainDataY;
            for (int i = 0; i < trainDx.rows(); i++){
                  file << trainDy(i) << " ";
                  for (int j = 0; j < trainDx.cols(); j++){
                        file << j + 1 << ":" << trainDx(i, j) << " ";
                  }
                  file << '\n';
            }

      }
```

Dieser Schritt ist notwendig, damit das mitgelieferte Kommandozeilenprogramm der LIBSVM einen Trainingsdatensatz für das Training und einen Testdatensatz zum Prädizieren in Form einer Textdatei hat. Das Ergebnis sehen wir in Abbildung 42

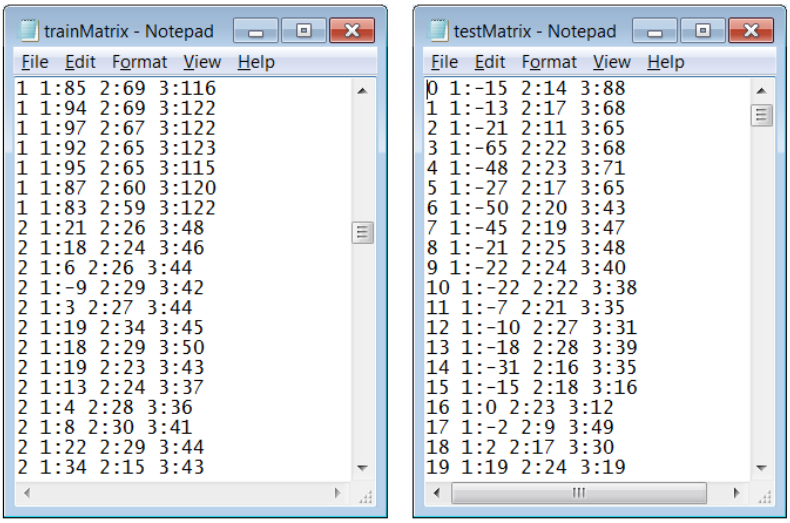


Abbildung 42: Konvertierte Trainings- und Testmatrizen der Bilder in Textdateiformat

Valide Eingabeformate des Trainings- und Testdatensatzes für das mitgelieferte Programm der LIBSVM

Jetzt wird im Validierungsprogramm der LIBSVM nochmals die svm-train Funktion aufgerufen, diesmal mit eingegebenen SVM-Parametern und dem Pfad zur in Abbildung 42 erstellten Trainingsmatrix (trainMatrix.txt) (siehe Abbildung 43)

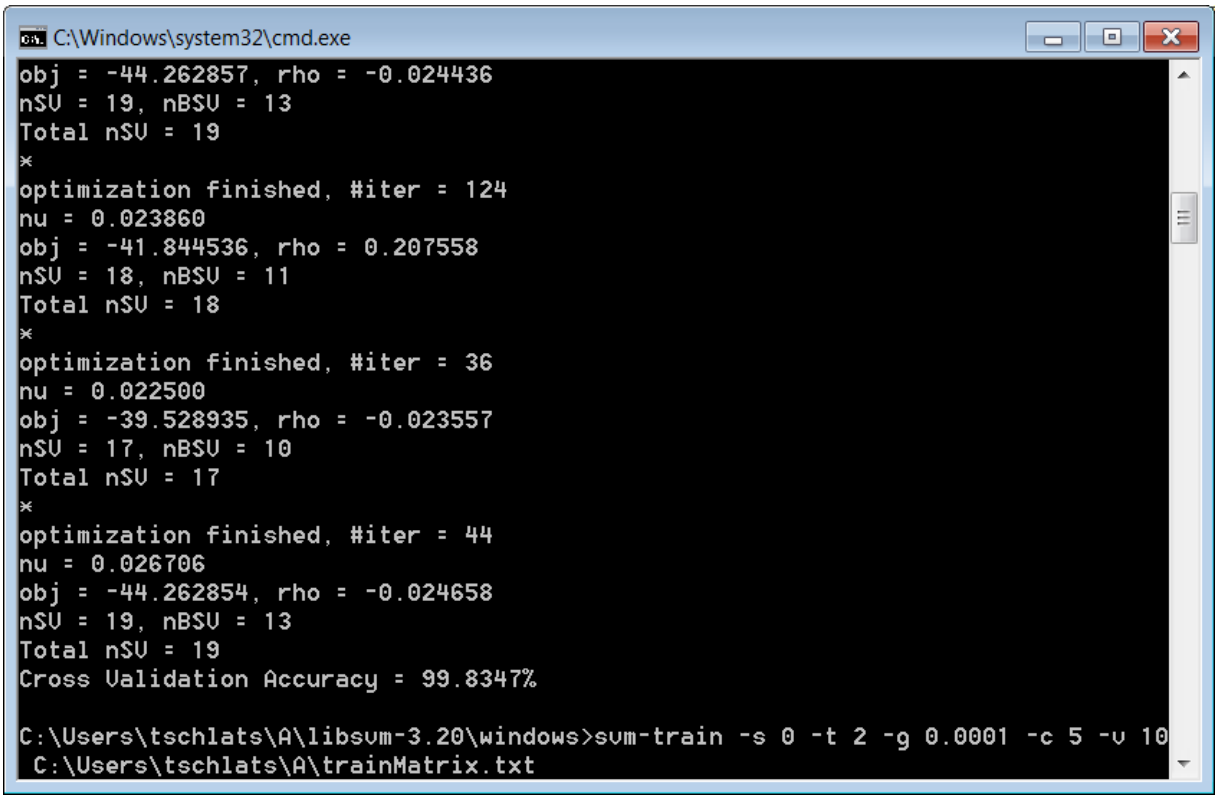


Abbildung 43: Validierungsprogramm der LIBSVM mit Kreuzvalidierung

Bei den gewählten Parametern erreichen wir eine Accuracy von 99,83%, dies ist ein sehr gutes Ergebnis, um im nächsten Schritt mit dem Training anzufangen. Man sollte immer bedenken, dass die Cross Validation Accuracy möglichst hoch sein soll, in diesem Fall bei mindestens 95%. Wenn zum Beispiel ein Patient ein Tumorvolumen von 80% aufweist und meine berechnete Cross Validation Accuracy bei 80% liegt, so kann es sein, dass gerade diese 80% nur Tumor entsprechen. Aus diesem Grunde würde man in so einem Fall mindestens eine Accuracy von 98% wählen, um gesundes und krankes Gewebe richtig zu klassifizieren. Ist bei der Parametereingabe die Cross-Validierung aktiviert (also wird der Parameter –v gesetzt), so wird nur eine Parameteroptimierung durchgeführt. Wenn der Parameter v nicht gesetzt wird, so trainiert das mitgelieferte Validierungsprogramm der LIBSVM ein Model.

Dieses Vorgehen der Parameteroptimierung wird jeweils immer dann durchgeführt, wenn die SVM für einen anderen Trainings- und Testdatensatz klassifizieren soll.

5.1.4 Versuchsergebnisse des Trainings

Nach der Datenintegration, der Auswahl der passenden Features und der Parameteroptimierung für den Real-Life-Test kann jetzt das Training der SVM durchgeführt werden. Dabei wird jeweils mit einem von drei Patientenkollektiven trainiert.

Patientenkollektiv 1:	Training: Patient A , Patient B	Test: Patient C
Patientenkollektiv 2:	Training: Patient B , Patient C	Test: Patient A
Patientenkollektiv 3:	Training: Patient A , Patient C	Test: Patient B

Zur Durchführung des Trainings wird die integrierte LIBSVM „MitkCLSVMClassifikation“ gestartet und berechnet eine Modeldatei für das Patientenkollektiv 1:

```
optimization finished, #iter = 649
nu = 0.734481
obj = -2168.860552, rho = -0.449960
nSV = 454, nBSV = 435
Total nSV = 454
4.85 core.mod.classification.clsvmclassification: File Saved ********************
**************************************
```

Abbildung 44: Berechnete Model-Datei mit der implementierten LIBSVM

Die SVM berechnete insgesamt (nSV) 454 Support Vektoren davon sind (nBSV) 435 Support Vektoren im berechneten Margin sogenannte bounded support vectors.

Parallel dazu hat die SVM in einem Verzeichnis „SVM-Train-Result“ die Modeldatei abgespeichert. (siehe Abbildung 45)

collection.xml | model.model

```
1  svm_type c_svc
2  kernel_type rbf
3  gamma 0.0001
4  nr_class 2
5  total_sv 454
6  rho -0.44996
7  label 1 2
8  nr_sv 231 223
9  SV
10 4.275524681303676 1:69 2:116
11 2.854458215470982 0:69 1:67 2:121
12 1.6864325877416 1:66 2:102
13 5 2:122
14 5 1:60 2:123
15 5 0:58 1:63 2:97
16 5 2:98
17 5 0:49 1:59 2:119
18 5 2:101
19 5 1:65 2:101
20 5 0:73 1:60 2:110
21 5 2:100
22 5 0:84 1:68 2:96
```

Abbildung 45: Ausgabe der Modeldatei im Real-Life-Test

Die Modeldatei zeigt die gewählten SVM Parameter, die nicht den Standardparameterwerten entsprechen. Anschließend ist die Gesamtanzahl der Support Vektoren (total_sv) 454 aufgeführt. Diese ist äquivalent zu Abbildung 44. Zu sehen sind auch die jeweiligen Supportvektoren einer Klasse. Z.B Label 1 mit 231 SV und Label 2 mit 223 SV. Ganz unten aufgelistet sind die einzelnen Support Vektoren (SV). Auch der Bias mit rho = -0.44996 ist identisch mit dem Wert in Abbildung 44.

Dieser Vorgang des Trainings wird für Patientenkollektiv 2 und 3 wiederholt. Die jeweilige Parameteroptimierung erfolgt analog wie in Kapitel 5.1.3 beschrieben.

Abbildung 46:
Berechnete Modeldateien für das Patientenkollektiv 2 links und 3 rechts

model.model

```
svm_type c_svc
kernel_type rbf
gamma 0.0006
nr_class 2
total_sv 652
rho -0.0460749
label 1 2
nr_sv 312 340
SV
4
4 0:128 1:74 2:112
4
4 2:106
2.764611961970131 0:151 1:74 2:115
4
2.31404849460073 1:59 2:107
3.912835169860719 0:109 1:56 2:113
4
0.7459599752349753 0:146 1:68 2:101
4
4
2.787957259903982 2:107
4 1:65 2:117
1.6564243067426 0:122 1:58 2:134
```

model.model

```
svm_type c_svc
kernel_type rbf
gamma 0.0006
nr_class 2
total_sv 533
rho -0.022704
label 1 2
nr_sv 254 279
SV
5
5 0:128 1:74 2:112
5
5 2:106
4.008889350200446 0:151 1:74 2:115
5
3.062156534732308 1:59 2:107
5 0:109 1:56 2:113
5
0.4296426474927093 0:146 1:68 2:101
5
5
2.271874899766624 2:107
3.536762629213863 1:65 2:117
1.708228260998727 0:122 1:58 2:134
```

5.1.5.1 Versuchsergebnisse des Prädizierens

Im letzten Schritt des Real-Life-Tests geht es darum, mithilfe der berechneten Modeldatei das Tumorvolumen in der Leber zu prädizieren. (siehe Kapitel 3.1) Damit dies möglich ist, wird die Modeldatei aus Kapitel 5.1.4 benötigt. Bei erneuter Ausführung des Programmes „MitkCLSVMClassification“ werden die Ergebnisbilder prädiziert und in einem Verzeichnis „SVM-Predict-Result“ gespeichert. (siehe Abbildung: 48)

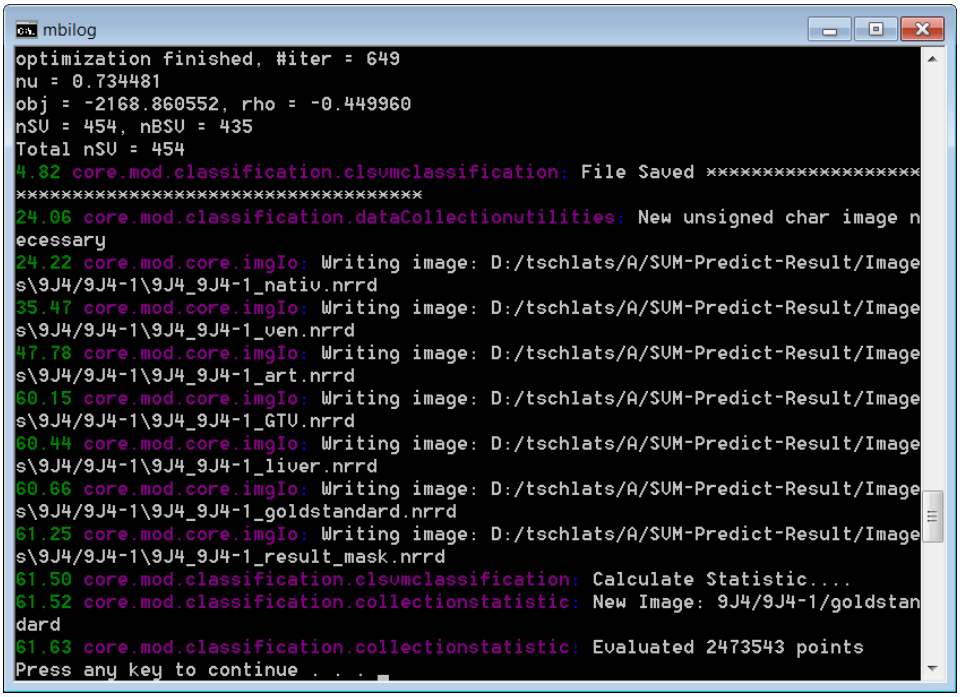


Abbildung 47: Ergebnis der Prädiktion der entwickelten LIBSVM

Angezeigt werden die berechnete Modeldatei sowie die Pfade, in denen die prädizierten Bilder gespeichert werden. Ganz unten ist zu sehen, dass auch Ergebnisstatistiken berechnet wurden.

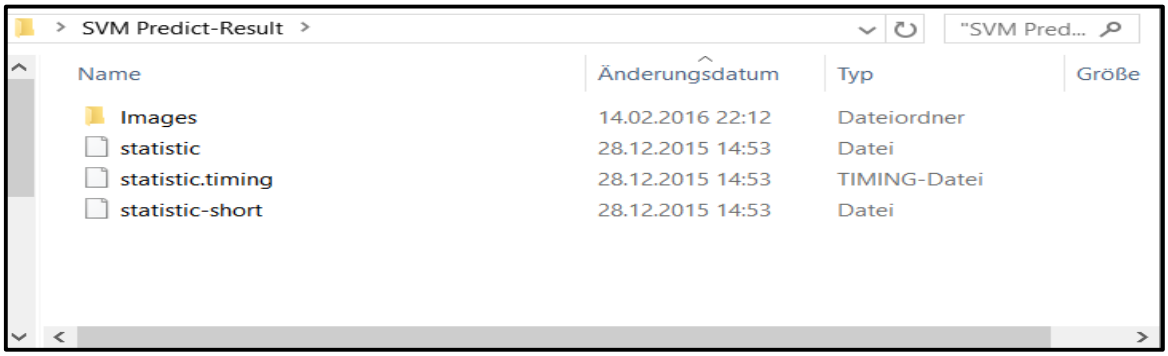
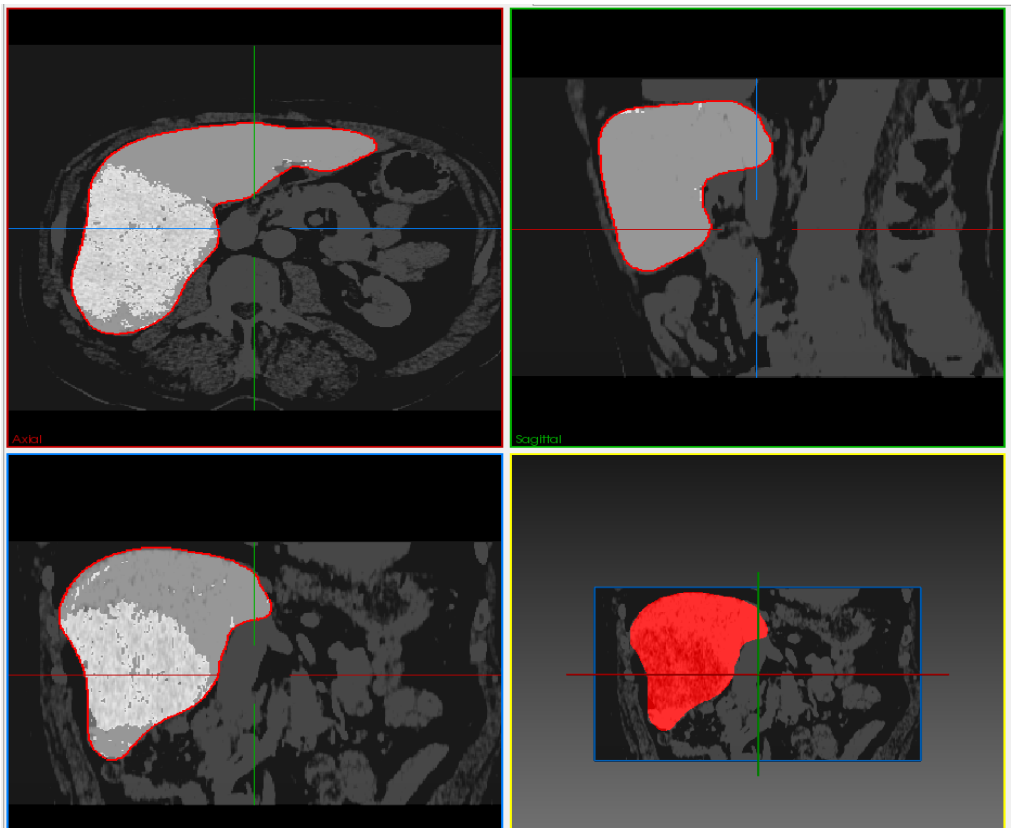


Abbildung 48: Speicherung der Prädiktionsergebnisse der SVM

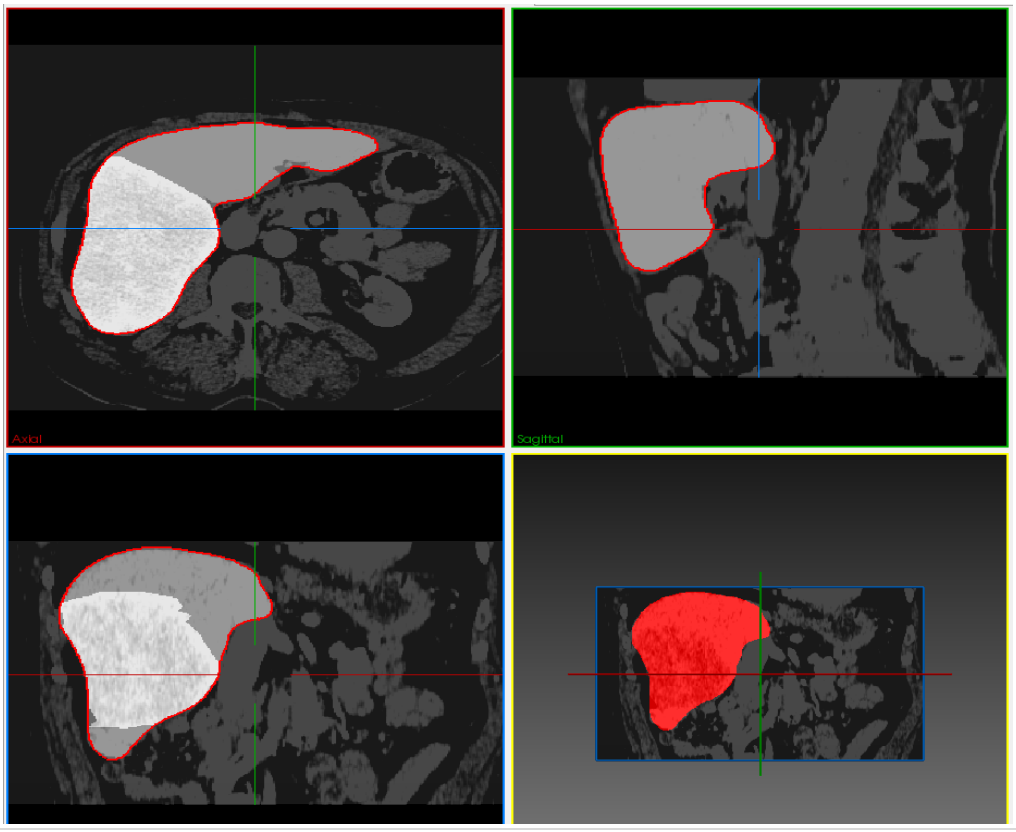
Verzeichnis, in dem die SVM im Ordner „Images“ die prädizierten Ergebnisbilder speichert. Zusätzlich werden noch drei Ergebnisstatistiken gespeichert.

Abbildung 49: Ergebnis der Prädiktion des Patientenkollektives 1

A Prädiktionsergebnis der SVM (Patient C)



B Goldstandard des prädizierten Ergebnisses (Patient C)



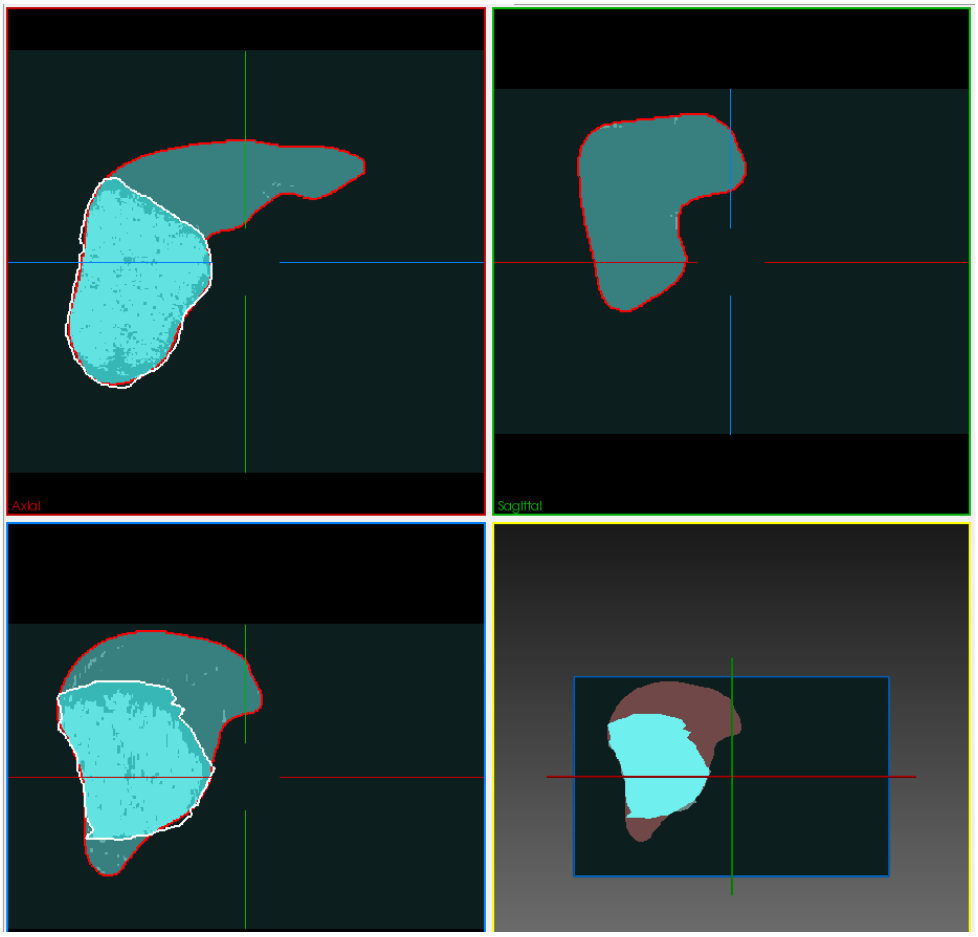
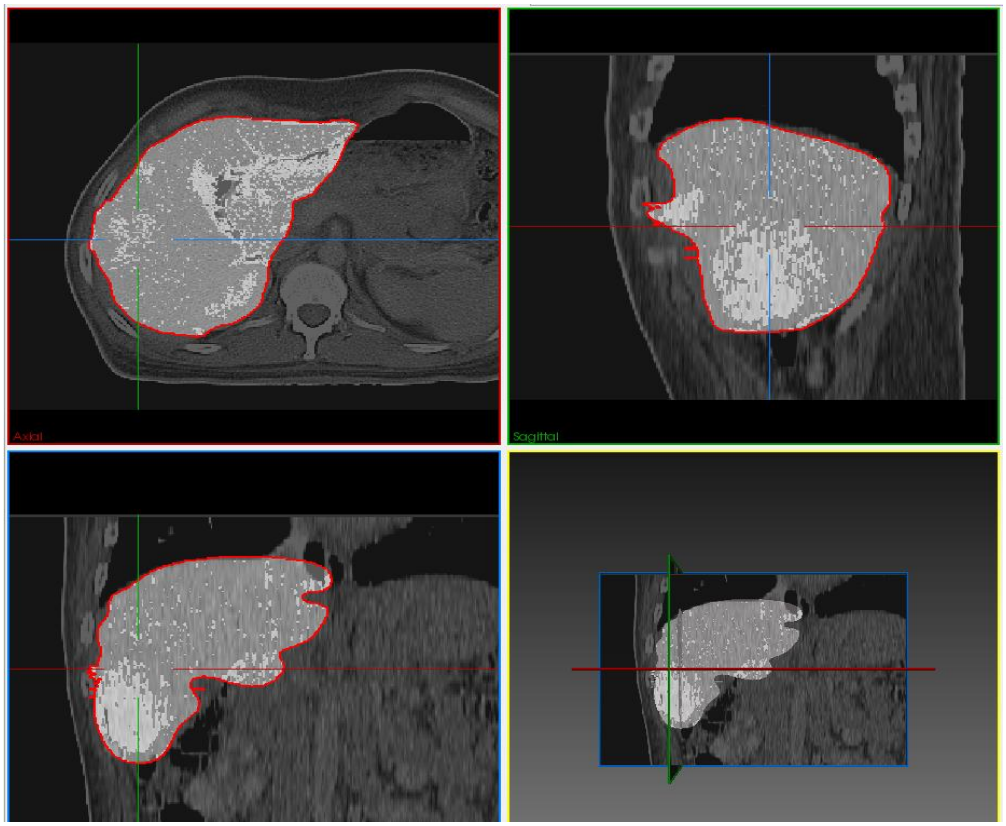


Abbildung 50: Überlagerungsbild des Patientenkollektives 1

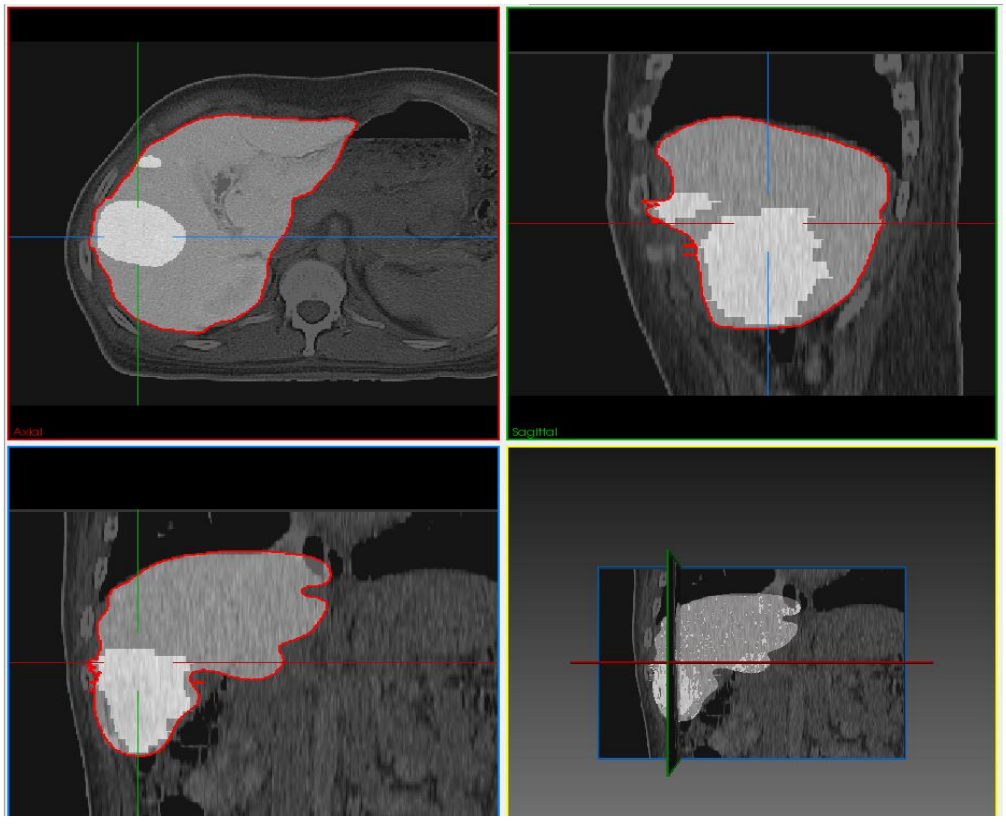
In Abbildung 49 a) ist das Klassifikationsergebnis der SVM beim Patienten C dargestellt. Abbildung 49 b) zeigt den Goldstandard, also das optimale Ergebnis, welches durch die Klassifikation erreicht werden sollte. Der Goldstandard entspricht einer Segmentierung der Leber mit Tumorvolumen, welches von einem Arzt manuell segmentiert worden ist. Die Güte des Klassifikationsergebnisses misst sich also am definierten Goldstandard. Abbildung 50 visualisiert eine Überlagerung der tatsächlich klassifizierten Lebertumorsegmentierung (blau eingefärbt) mit dem Goldstandard der Tumorsegmentierung (weiße Umrandung in den Rendering-Windows) des Arztes. So lässt sich am besten abschätzen, wie gut das tatsächliche Prädiktionsergebnis im Vergleich zum Goldstandard ist. Aufgrund des Ergebnisses von Abbildung 50 kann man deutlich erkennen, dass die entwickelte SVM visuell beurteilt ein hinreichend gutes Klassifikationsergebnis erreicht hat. Dies zeigt sich, indem das Volumen der finalen Segmentierung in hohem Maße mit dem Volumen des Goldstandards übereinstimmt. Die präzise Angabe der Ähnlichkeit beider Segmentierungen wird anhand des Dice-Koeffizienten, einer statistischen Kenngröße zum Vergleich der Ähnlichkeit zweier Segmentierungen, festgestellt. (siehe Kapitel 5.1.5.2)

Abbildung 51: Ergebnis der Prädiktion des Patientenkollektives 2

A Prädiktionsergebnis der SVM (Patient A)



B Goldstandard des prädizierten Ergebnisses (Patient A)



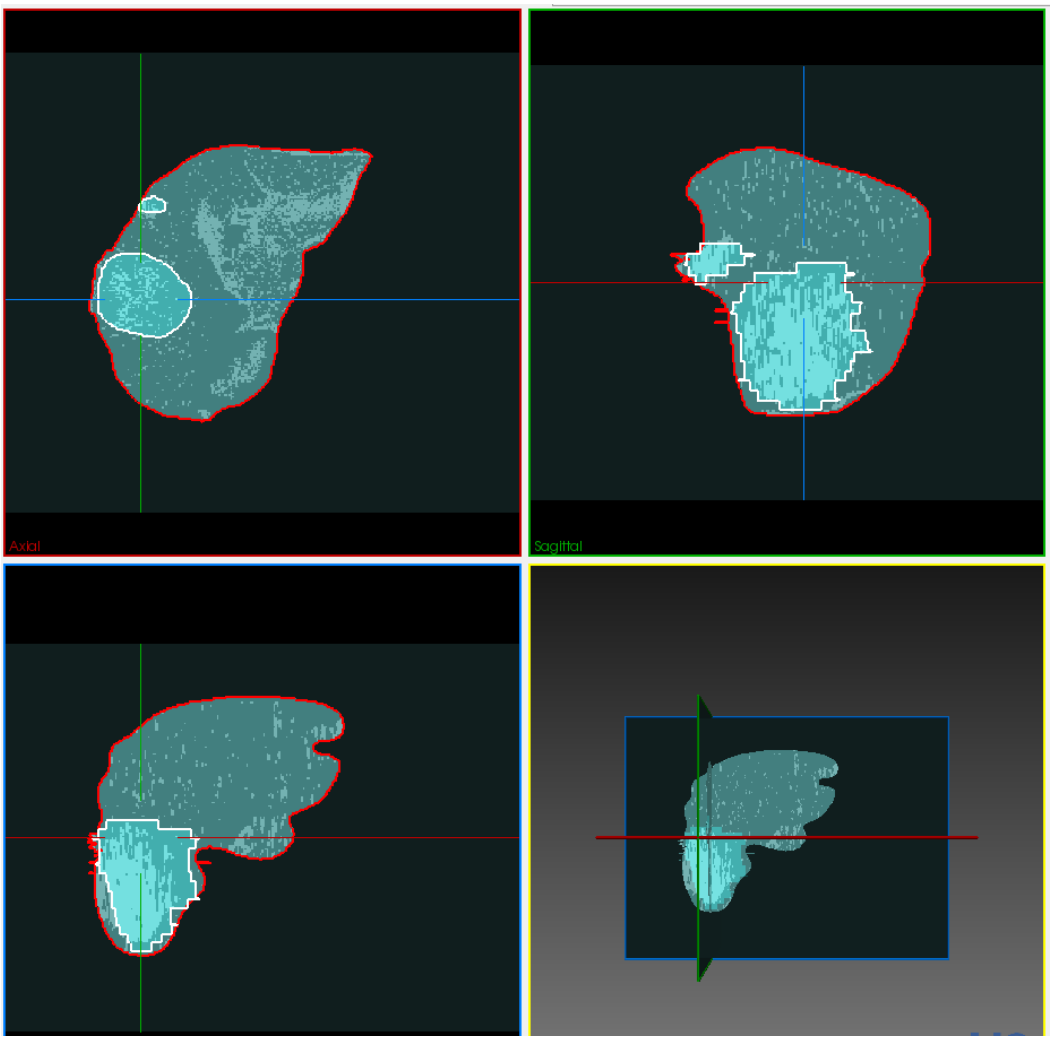
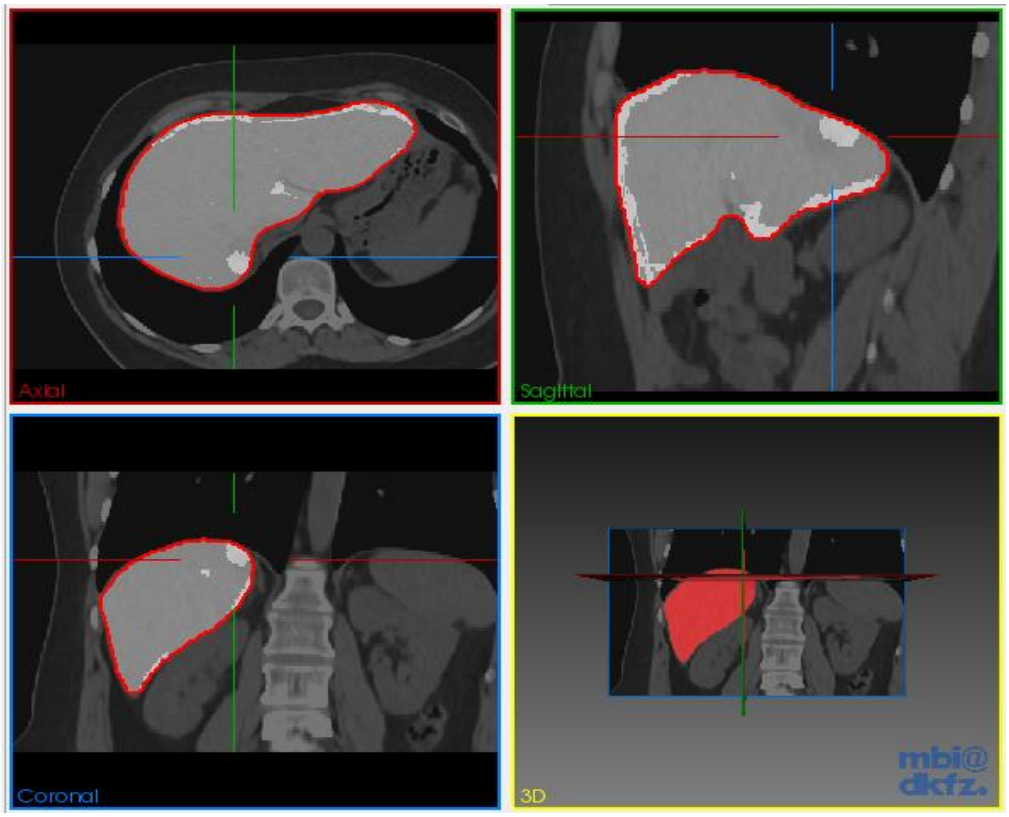


Abbildung 52: Überlagerungsbild des Patientenkollektives 2

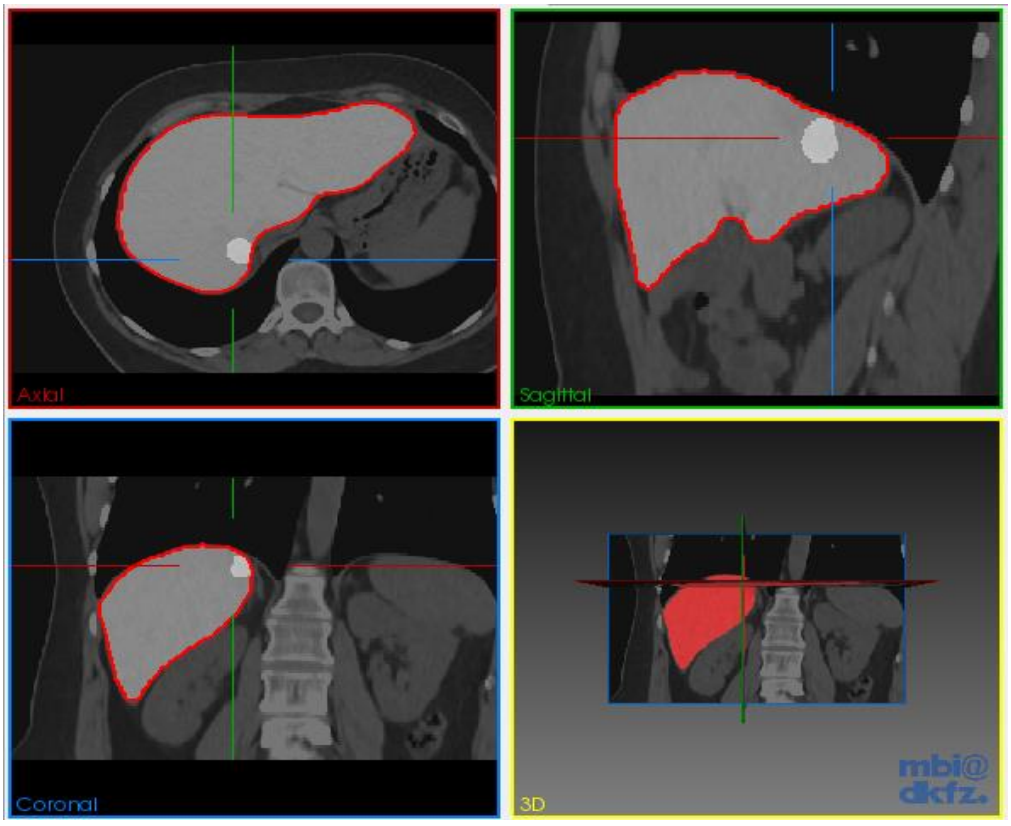
Analog zur Beschreibung des Falles von Patient C entspricht Abbildung 51 a) dem Prädiktionsergebnis der SVM beim Patienten A und Abbildung 51 b) dem Goldstandard des Patienten A. In der Abbildung 52 kann man erkennen, dass der Goldstandard in der transversalen Ebene (Abbildung 52 oben links) unzureichend richtig klassifiziert wurde, da fast keine hellblauen Punkte in den Bereichen des Goldstandards (weiße Umrandung) vorhanden sind. Zusätzlich fällt auf, dass in allen drei orthogonalen Ebenen ein starkes Bildrauschen vorhanden ist. Der prädizierte Bereich in der transversalen Ebene außerhalb des Goldstandards (s-förmige weiße Linie) ist eindeutig eine Fehlklassifikation. Diese Fehlklassifikation kann durch eine Vene zustande gekommen sein, die sich während der Atembewegung des Patienten hervorhebt und der Klassifikator diese fälschlicherweise als Tumorregion klassifiziert hat. In der sagittalen Ebene (Abbildung 52 rechts oben) und in der coronalen Ebene (Abbildung 52 links unten) prädizierte die SVM wesentlich besser, gut zu sehen an den hellblauen Punkten innerhalb des weißen Bereiches. Die statistische Auswertung dieser Prädiktion findet sich ebenfalls in Kapitel 5.1.5.2.

Abbildung 53: Ergebnis der Prädiktion des Patientenkollektives 3

A Prädiktionsergebnis der SVM (Patient B)



B Goldstandard des prädizierten Ergebnisses (Patient B)



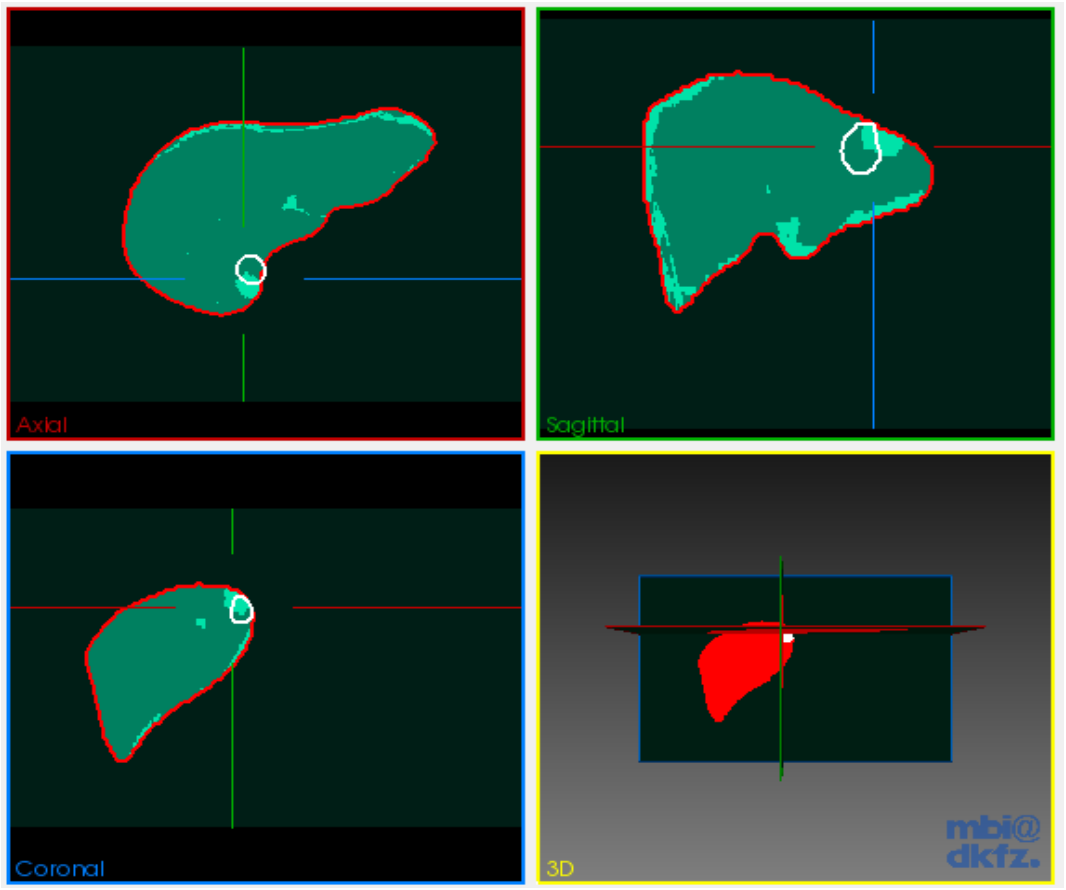


Abbildung 54: Überlagerungsbild des Patientenkollektives 3

Das letzte Segmentierungsergebnis (Patient B) des Real-Life-Tests schneidet sowohl visuell (in Abbildung 54) als auch statistisch (in Kapitel 5.1.5.2) betrachtet als schlechtestes Klassifikationsergebnis ab. Bei Betrachtung von Abbildung 54 sieht man in allen Rendering-Windows, dass das prädizierte Tumorvolumen der SVM nur partiell oder gar nicht im definierten Goldstandard (weiße Umrandung) liegt. Zudem kann man in allen Ebenen am Rande der Lebersegmentierung (rot visualisiert) partiell auftauchend hellblaue Tumorregionen sowie jeweils eine einzelne Tumorklassifikation in der Mitte der Schichten beobachten. Dabei handelt es sich eindeutig um Fehlklassifikationen.

5.1.5.2 Statistische Auswertung der Prädiktionsergebnisse

Eine essentielle Kenngröße für die Auswertung der prädizierten Segmentierungsergebnisse ist der so genannte Dice-Koeffizient. Dieser beschreibt die Ähnlichkeit zwischen zwei Segmentierungen und lässt sich wie folgt berechnen:

$$D(A,B) = \frac{2 \cdot |A \cap B|}{|A| + |B|} \quad (13) \qquad [HH\ 09]$$

Dabei entsprechen A und B den Voxelelementen der beobachteten Segmentierungen. Liegt der Dice-Koeffizient bei 1, so stimmen die Segmentierungen A und B in hohem Maße überein (zu 100%). Bei einem Koeffizienten, der asymptotisch gegen 0 konvergiert, ist von keiner Ähnlichkeit der Segmentierungen A und B auszugehen. [HH 09]

Eine weitere Kenngröße zum Vergleich von Segmentierungen ist der Jaccard-Koeffizient.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (14) \qquad [HH\ 09]$$

Genauso wie der Dice-Koeffizient nimmt auch der Jaccard-Koeffizient die Schnittmenge beider Segmentierungen $|A \cap B|$ als Kenngröße. Jedoch unterscheiden sich beide Koeffizienten aufgrund „der Art der Normierung“. [HH 09] Ein Jaccard-Koeffizient des Werts 1 entspricht ebenfalls einer identischen Ähnlichkeit beider Segmentierungen. [HH 09]

Mithilfe der eingeführten Koeffizienten, einer Statistik und den Ergebnisbildern möchte ich die prädizierten Ergebnisse des Real-Life-Tests auswerten. Die nachfolgenden Abbildungen 55 bis 57 zeigen die jeweiligen Statistiken an, welche die implementierte SVM nach jeder Klassifikation automatisch berechnet. Die aufgeführten Klassen 0 bis 2 geben die Labelwerte an, d.h. in der Klasse 0 sind alle Voxel, die nicht zum Tumorgewebe und auch nicht zum gesunden Lebergewebe zählen, also die Voxel des Hintergrundes der Segmentierung. In der Klasse 1 befinden sich Voxel, die gesundes Lebegewebe repräsentieren. Die Klasse 2 beinhaltet die Voxelregionen, welche dem Lebertumor entsprechen. Mengenmäßig betrachtet kann man annehmen, dass die Menge A den Voxelelementen der prädizierten Segmentierung und die Menge B die der Goldstandardsegmentierung entspricht. Bei Betrachtung der Statistiken (Abbildung 55 bis 57) erkennt man, dass sich der Dice-Koeffizient asymmetrisch verhält. Dies bedeutet, die TP (true positive) Werte der Klasse 1 sind fast identisch mit den TN (true negative) Werten der Klasse 2 und umgekehrt. Genauso gilt, dass die FP (false positive) Werte der Klasse 1 äquivalent mit den FN (false negative) Werten der Klasse 2 sind, auch für den umgekehrten Fall. Der Mean-Wert in den nachfolgenden Statistiken gibt das arithmetische Mittel an. W-Mean bezieht sich auf den Mittelwert mit Berücksichtigung der verwendeten Gewichte bei der Klassifikation. In unserem Fall wird keine Klassifikation mit sogenannten „gewichteten Werten“ (weighted-Values) durchgeführt. Man kann die „Gewichte“ beim Klassifikationsprozess verwenden, dafür gibt es einen extra SVM-Parameter.

Der Dice-Koeffizient berechnet sich für den Patienten C gemäß Abbildung 55 wie folgt:

Merkmal	A	$\overline{A}$	$\sum$
B	$TP(A \cap B)$	$FP(\overline{A} \cap B)$	B
$\overline{B}$	$FN(A \cap \overline{B})$	$TN(\overline{A} \cap \overline{B})$	$\overline{B}$
$\sum$	A	$\overline{A}$	N

$$Allgemein: \; D(A,B) = \frac{2\,TP}{TP + FN + TP + FP} \quad (15)$$

$$Klasse\ 1: \; D(A,B) = \frac{2 * 1583809}{1583809 + 33135 + 1583809 + 266700} = 0{,}91353$$

$$Klasse\ 2: \; D(A,B) = \frac{2 * 589898}{589898 + 266656 + 589898 + 33136} = 0.79738$$

$$Klasse\ 2: \; Accuracy: \frac{589898 + 1583853}{589898 + 1583853 + 33136 + 266656} = 0{,}8788$$

Damit ergibt sich eine Übereinstimmung des Anteils von, klassifiziertem gesundem Lebergewebe mit dem Goldstandard von 91,353%. Beim prädizierten Tumorgewebe wird eine Ähnlichkeit beider Segmentierungen von 79,738% berechnet. Aus Erfahrung kann man sagen, dass ein Dice-Koeffizient von mind. 70% einem guten Klassifikationsergebnis entspricht. Erwartungsgemäß bestätigt der ermittelte Dice-Koeffizient des Patienten C das visuelle Ergebnis aus Abbildung 50 im Kapitel 5.1.5.1. Auch die Cross-Validation-Accuracy mit 87,88% richtig klassifizierter Merkmale bestätigt das gute Klassifikationsergebnis im Fall des Patienten C.

========== Image 0 ==========
Image ID : 9J4/9J4-1/goldstandard

Class	DICE	Jaccard	Sensitivity	Specificity	TP	TN	FP	FN
0	0	0	0	1	0	2473498	0	45
1	0.91353	0.84082	0.97951	0.68865	1583809	589899	266700	33135
2	0.79738	0.66304	0.68869	0.97951	589898	1583853	33136	266656
Mean	0.5703	0.50129	0.55607	0.88939	724569	1549083	99945	99945
W-Mean	0.87329	0.77924	0.87878	0.78938	0	1	0	0
Compl.	x	x	0.87878	x	2173707	0	299836	299836

Abbildung 55: Ergebnisstatistik zur Prädiktion des Patienten C

Die Berechnung des Dice-Koeffizienten für Patient A lehnt sich an die Statistik aus Abbildung 56 an und lässt sich aus Formel 15 wie folgt berechnen:

$$Klasse\ 1:\ D(A,B) = \frac{2 * 1373136}{1373136 + 178280 + 1373136 + 93771} = 0{,}90987$$

$$Klasse\ 2:\ D(A,B) = \frac{2 * 69248}{69248 + 93771 + 69248 + 178280} = 0.33735$$

$$Klasse\ 2:\ Accuracy:\ \frac{69248 + 1373136}{69248 + 1373136 + 178280 + 93771} = 0{,}8414$$

Als statistisches Ergebnis wird eine Übereinstimmung der Prädiktion des gesunden Lebergewebes mit dem Goldstandard beim Patienten A von 90,987% erreicht. Der Dice-Koeffizient liegt beim prädizierten Tumorgewebe von Patient A nur bei 33,375%, was schon in Kapitel 5.1.5.1 (Abbildung 52) ersichtlich war. Das bedeutet, nur ein Drittel des gesamten Tumorvolumens wurde richtig klassifiziert. Trotz einer hohen Accuracy von 84,14% hat die entwickelte SVM ein unterdurchschnittliches Klassifikationsergebnis erreicht. Die hohe Accuracy ist in diesem Fall dadurch zu erklären, dass über 90% des gesunden Lebergewebes richtig klassifiziert wurde.

========== Image 0 ==========

Image ID : 73L/73L-1/goldstandard

Class	DICE	Jaccard	Sensitivity	Specificity	TP	TN	FP	FN
0	0	0	0	1	0	1714435	0	0
1	0.90987	0.83464	0.88509	0.42478	1373136	69248	93771	178280
2	0.33735	0.2029	0.42478	0.88509	69248	1373136	178280	93771
Mean	0.41574	0.34584	0.43662	0.76996	480794	1052273	90683	90683
W-Mean	0.85543	0.77457	0.84132	0.46855	0	1	0	0
Compl.	x	x	0.84132	x	1442384	0	272051	272051

Abbildung 56: Ergebnisstatistik zur Prädiktion des Patienten A

Abschließend wird der Dice-Koeffizient für die Klassifikation des Patienten B berechnet, in Anlehnung an Abbildung 57:

$$Klasse\ 1:\ D(A,B) = \frac{2 * 1171236}{1171236 + 122972 + 1171236 + 6704} = 0{,}94755$$

$$Klasse\ 2:\ D(A,B) = \frac{2 * 1569}{1569 + 6665 + 1569 + 122977} = 0.023633$$

$$Klasse\ 2:\ Accuracy:\ \frac{1569 + 1171275}{1569 + 1171275 + 122977 + 6665} = 0{,}9005$$

Wie zu erwarten war, ist das prädizierte gesunde Lebergewebe mit einem Dice-Score von 94,755% sehr gut klassifiziert worden. Aber das Tumorgewebe des Patienten B ist mit einem Dice-Koeffizienten von 2,3633% (d.h. eine nur zweiprozentige Übereinstimmung mit Goldstandard) ungenügend von der SVM klassifiziert worden. Zwar liegt die Accuracy bei 90,05%, jedoch bezieht sich diese auf das richtig klassifizierte gesunde Lebergewebe. So bestätigt der berechnete Dice-Koeffizient die Fehlklassifikation, die in Abbildung 54 (Kapitel 5.1.5.1) schon zu sehen war.

========== Image 0 ==========

Image ID : DQA/DQA-1/goldstandard

Class	DICE	Jaccard	Sensitivity	Specificity	TP	TN	FP	FN
0	0	0	0	1	0	1302442	0	44
1	0.94755	0.90032	0.90498	0.19014	1171236	1574	6704	122972
2	0.023633	0.011958	0.19055	0.90498	1569	1171275	122977	6665
Mean	0.32373	0.30409	0.36518	0.69837	390935	825097	43227	43227
W-Mean	0.94167	0.89467	0.90044	0.19469	0	1	0	0
Compl.	x	x	0.90044	x	1172805	0	129681	129681

Abbildung 57: Ergebnisstatistik zur Prädiktion des Patienten B

5.1.5.3 Integration von zusätzlichen Features

Die bisher prädizierten Segmentierungsergebnisse können sich durch die Hinzunahme von zusätzlichen Merkmalen in Form von gaußgeglätteten Bildern verbessern, müssen es aber nicht zwingend. Zur Erstellung von gaußgeglätteten Bildern wird das vom DKFZ entwickelte Programm „CLVoxelFeatures“, wie in Kapitel 5.1.2 beschrieben, verwendet. Für den Patienten C wurde die Methode „Difference of Gaussian“ (DoG) des Programmes genutzt. Dabei wurden iterativ die Bildmodalitäten arteriell, venös und nativ mit einer Glättungsvarianz von $\sigma = 1.5$ und der Filterart DoG dem Programm übergeben. Die geglätteten Bilder finden sich im Speicherverzeichnis des Patienten C wieder. (siehe Abbildung 58)

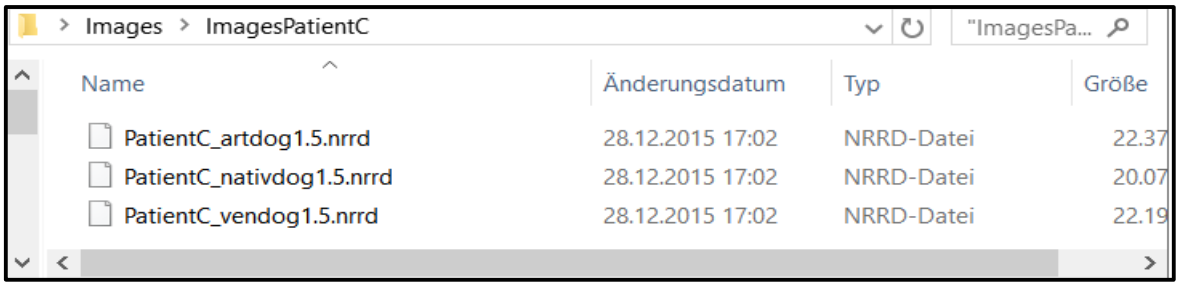


Abbildung 58: Speicherverzeichnis für Patient C

Damit die entwickelte SVM diese Bilder bei der Klassifikation verwenden kann, werden diese in die bereits existierende „Collection.xml“ eingefügt. Beim erneuten Starten der SVM werden für den Patienten C jetzt folgende Segmentierungen prädiziert: (siehe Abbildung 59)

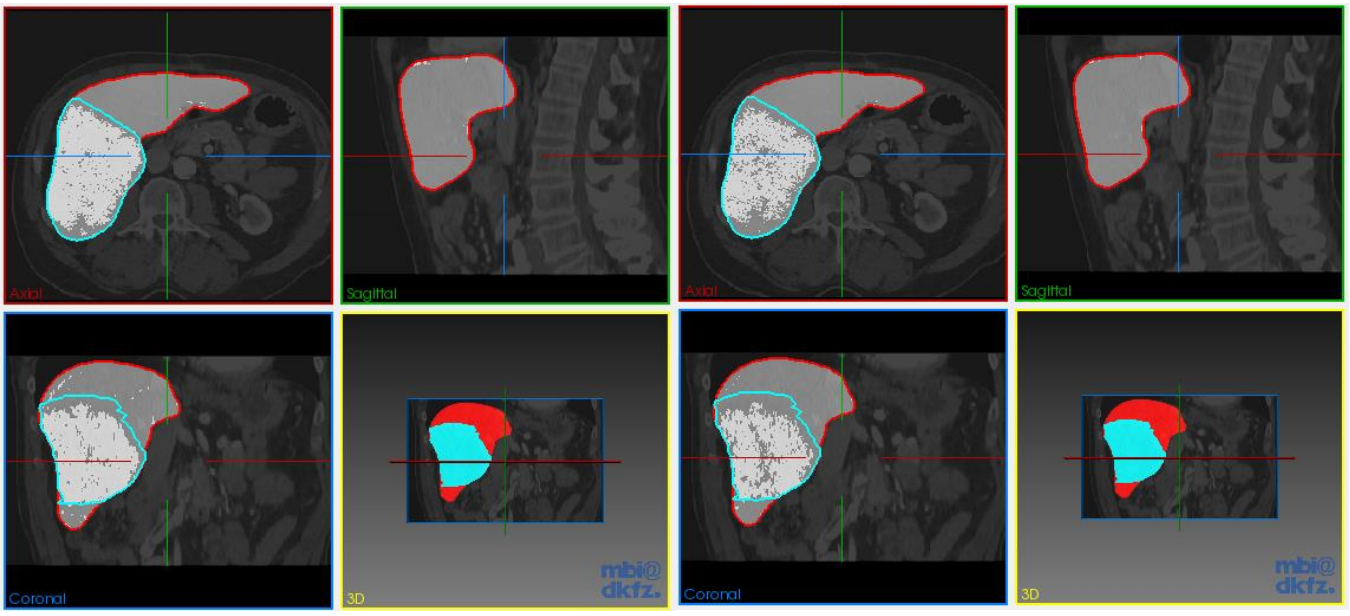


Abbildung 59: SVM Prädiktion vor und nach Glättung für den Patienten C

Links abgebildet ist das ursprüngliche Prädiktionsergebnis vor der Glättung, rechts sieht man das Klassifikationsergebnis nach einer DoG Glättung.

Festzustellen ist, dass nach einer DoG-Glättung das Prädiktionsergebnis der SVM stärker vom Goldstandard abweicht als das vorher der Fall war. (vgl. Abbildung 59) Dies bestätigt auch der neu berechnete Dice-Koeffizient. (siehe Abbildung 60) Jetzt wird ein Dice-Score beim gesunden Lebergewebe von 88,39% und beim Tumorgewebe von 67,9% erreicht. Vergleicht man den Dice-Score vor und nach der Glättung, so stellt man eine Verschlechterung beim gesunden Lebergewebe von 2,96% und beim Tumorgewebe von 11,84% fest. Das könnte daran liegen, dass die DoG-Glättung in den arteriellen, venösen und nativen Bildaufnahme andere Kanten lokalisiert und detektiert hat, die zum Tumorvolumen gehören, als dies beim Goldstandard der Fall war. Die Glättung hat somit einheitlich zusammenfassende, dunkle Grauwertbereiche Tumorgewebe zugeordnet. Der Mediziner hat bei der Segmentierung des Goldstandards aber noch andere Grauwerte als die von DoG-Filter detektierten, als Tumorregion definiert.

========== Image 0 ==========

Image ID : 9J4/9J4-1/goldstandard

Class	DICE	Jaccard	Sensitivity	Specificity	TP	TN	FP	FN
0	0	0	0	1	0	2473498	0	45
1	0.88393	0.792	0.99303	0.52085	1605679	446158	410441	11265
2	0.67909	0.51411	0.52087	0.99303	446157	1605723	11266	410397
Mean	0.52101	0.43537	0.50464	0.83796	683945	1508459	140569	140569
W-Mean	0.81298	0.69575	0.82951	0.68437	0	1	0	0
Compl.	x	x	0.82951	x	2051836	0	421707	421707

Abbildung 60: Statistik nach Prädiktion der Glättung von Patient C

Bei genauerer Betrachtung des Prädiktionsergebnises von Patient A (Abbildung 61) nach einer einfachen Gaußglättung fällt auf, dass das Segmentierungsergebnis des Patienten stark verrauscht ist. Zur Entfernung des Rauschens wird ein einfacher Gaußglättungsfilter („Gaussian“) verwendet. Auch hier wird dem Programm „CLVoxelFeatures“ jeweils eine Modalität (arteriell, venös und nativ) sowie eine Glättungsvarianz von $\sigma = 1.5$ mit der Filterart „Gaussian“ übergeben. Als Ergebnis werden die geglätteten Bilder im Verzeichnis „Image-PatientA“ gespeichert. Anschließend werden auch diese Bilder in die bestehende „Collection.xml“ integriert. Nach erneutem Ausführen der SVM werden folgende Labelbilder generiert. (siehe Abbildung 61) Bei Betrachtung von Abbildung 61 stellen wir fest, dass durch die Glättung das vorher bestehende Bildrauschen komplett entfernt wurde. Jedoch sieht man auch, dass vorher richtig prädiziertes Tumorvolumen durch die Glättung partiell entfernt wurde. (vgl. Goldstandardbereich cyan gefärbt mit den enthaltenen weiß prädizierten Tumorvoxeln). Ein möglicher Grund dafür ist, dass der Gaußfilter Tumorvoxel als Rauschen detektiert hat und dieses entfernte. Die beigefügte Statistik aus Abbildung 62 bestätigt meine Vermutung. Der Dice-Koeffizient der Tumor-segmentierung hat durch die Filterung um 4,96% abgenommen und beträgt jetzt 28,78%. Die Entfernung des Rauschens und somit die Erhöhung des Anteils von richtig prädizierten gesundem Lebergewebe können wir anhand des Dice-Koeffizienten der gesunden Lebersegmentierung beobachten. Dieser steigt von 90,99% (vor der Glättung) auf 94,14% (nach der Glättung).

Somit führte der Einsatz des Gaußfilters einerseits zu einer Verschlechterung des prädizierten Tumorvolumens, andererseits aber zu einer Verbesserung des klassifizierten gesunden Lebergewebes.

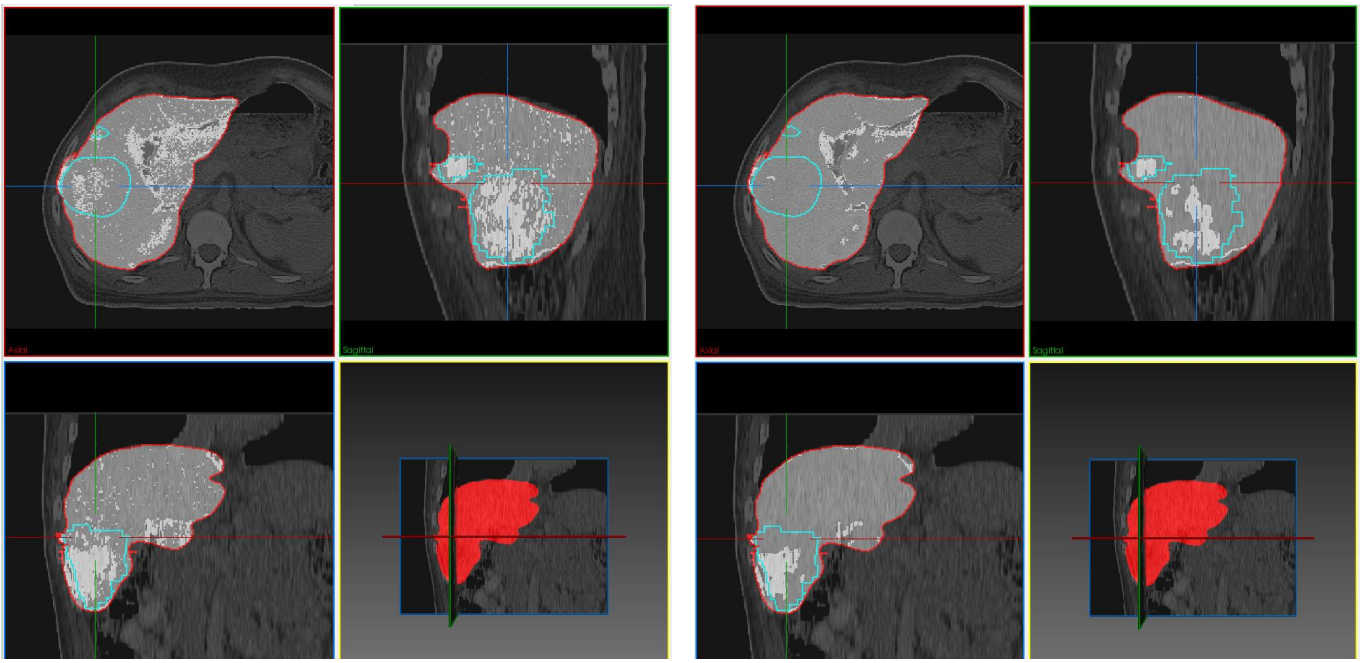


Abbildung 61: SVM Prädiktion vor und nach Glättung für den Patienten A

Links abgebildet ist das ursprüngliche Prädiktionsergebnis vor der Glättung, rechts sieht man das Klassifikationsergebnis nach einer Gaußglättung.

========== Image 0 ==========

Image ID : 73L/73L-1/goldstandard

Class	DICE	Jaccard	Sensitivity	Specificity	TP	TN	FP	FN
0	0	0	0	1	0	1714435	0	0
1	0.9414	0.88928	0.96122	0.23014	1491252	37517	125502	60164
2	0.28782	0.1681	0.23014	0.96122	37517	1491252	60164	125502
Mean	0.40974	0.35246	0.39712	0.73045	509589	1081068	61888	61888
W-Mean	0.87925	0.82071	0.8917	0.29965	0	1	0	0
Compl.	x	x	0.8917	x	1528769	0	185666	185666

Abbildung 62: Statistik nach der Prädiktion der Glättung von Patient A

Zusammenfassung des Real-Life-Tests

Zusammenfassend kann man sagen, dass beim Real-Life-Test mit den gegebenen Patientendatensätzen alles dabei war, von sehr guten Klassifikationsergebnissen (vgl. Patient C) über mittelgute (vgl. Patient A) bis hin zu sehr schlechten (vgl. Patient B). Man konnte auch sehen, dass die Verwendung von zusätzlichen Features das Klassifikationsergebnis beeinflusst. Leider ist keine Lebersegmentierung der SVM zu 100% mit dem Goldstandard äquivalent gewesen. Jedoch ist es möglich, mit der Technik der SVM Lebertumore trotz physiologischer Atembewegungen gut zu klassifizieren. Es kann natürlich zu morphologisch bedingten Fehlklassifikationen (vgl. Patient A) kommen, da beispielsweise der SVM einen Venengang als Lebertumor klassifiziert. Diese Fehlklassifikationen kann man jedoch als konstruktiven Ansporn verstehen, um die integrierte SVM zu verbessern. Der Real-Life-Test stellte somit einen tatsächlichen komplexen Anwendungsfall da, um den Sourcecode der entwickelten SVM, bei großen Datenmengen zu validieren.

6. Fazit und Ausblick

Im letzten Kapitel meiner Bachelorarbeit möchte ich die Ergebnisse des Real-Life-Tests (Kapitel 5) und die Implementierung der SVM kritisch diskutieren. Anschließend möchte ich einen Ausblick über die zukünftigen Einsatzgebiete der entwickelten SVM geben.

6.1 Ergebnisdiskussion

Das primäre Ziel der Integration einer LIBSVM in MITK ist erreicht worden, auch wenn es anfangs nicht leicht war, die SVM zum Laufen zu bringen. Ein Problem mit, dem ich mich länger beschäftigt hatte, bestand darin, dass in die trainierte Modeldatei nicht richtig hineingeschrieben worden ist. Der Grund war ein Fehler in der Wrapperklasse, welcher dafür sorgte, dass die vom Computer erkannten virtuellen Minuseinsen, die einen Zeilenumbruch in der Modeldatei erzwingen, falsch gesetzt worden sind. Dadurch hat die entwickelte SVM falsche Ergebnisse produziert. Mithilfe meiner selber implementierten Testklasse habe ich das Problem erkannt. Dabei bin ich sehr strukturiert vorgegangen. Zuerst schaute ich mir die Trainings- und die Testdaten an, um festzustellen, dass z.B. eine Voxelregion (arteriell, venös und nativ) nicht gleichzeitig einem Labelwert „Tumor“ (2) und „Nichttumor“ (1) zugeordnet ist. Dafür nutzte ich in meiner Testklasse den implementierten CSV-Writer, um die Daten in eine CSV-Datei zu exportieren. Anschließend konnte ich mir durch die tabellarische Form der Daten einen guten Überblick verschaffen. Nach Feststellung der Datenkonsistenz, validierte ich die SVM-Parameter mit dem vorgestellten Kreuzvalidierungsverfahren. Dabei stellte ich fest, dass die erreichte Accuracy maximal bei 30% lag, obwohl ich nach Kapitel 2.3.3.3 logisch betrachtet, alle sinnvollen Parameterkonfigurationen ausprobiert hatte. Danach trainierte ich mit Hilfe meiner Testklasse eine Modeldatei und schaue nach, wie in diese Model-Datei hineingeschrieben wurde. Auffällig war, dass die ganzen Voxel in der Trainingsmaske in die erste Zeile des Modells geschrieben wurden. Mit zunehmenden Zeillenindex, wurde immer das letzte Voxel der vorherigen Zeile nicht in die aktuelle Zeile der Datei geschrieben. Dadurch erkannte ich, dass die Daten nicht richtig in die Model-Datei geschrieben wurden und löste das Problem, indem ich, an der entsprechenden Stelle in der Wrapperklasse, einen Schleifenparameter mit i + 1 in der Matrix inkrementierte. Da die Klassifikation sehr datenabhängig ist, kann man derartige Fehler nur schwer erkennen, obwohl das Programm funktioniert.

Der automatisierte Test, welcher speziell für die entwickelte Software geschrieben wurde, hat sehr gute Ergebnisse geliefert. Somit ist sichergestellt, dass die integrierte SVM richtig klassifizieren kann. Anhand des entwickelten Tests (in Kapitel 4) geht auch hervor, dass die entwickelte SVM für statistisch verteilte Datensätze generell besser klassifiziert als für nicht statistisch verteilte Daten. Das kann daran liegen, dass beispielsweise bei gaußnormalverteilten Datensätzen sowohl die Verteilungsfunktion (Gaußkurve) ähnlich ist als auch die der Entscheidungsfunktion (Gaußkurve) der SVM beim RBF-Kernel.

Sicher ist, wenn z.B. die Tumorvoxel innerhalb einer mehrdimensionalen Gaußnormalverteilung liegen und die Nichtumorvoxel außerhalb der Verteilungsfunktion definiert sind, so die SVM mit einem Gaußkernel die Voxel eindeutig klassifizieren kann, da die entsprechende Entscheidungsfunktion (Hyperebene) ebenfalls eine mehrdimensionale Gaußnormalverteilung ist und die Samples nicht vermischt sind.

Die entwickelte SVM kann zukünftig auch für alle anderen Tumorarten eingesetzt werden. Diese wurde so implementiert, dass der Klassifikator generell zwischen gesundem und krankem Tumorgewebe differenzieren und prädizieren kann. Die Ergebnisse des Real-Life-Tests zeigen jedoch auf, dass es noch reichlich Luft nach oben im Hinblick auf die Erreichung des Goldstandards gibt.

Ein weiteres Problem während der Entwicklung der SVM war es, dass zu wenig reale Patientendaten verfügbar waren. Insgesamt standen nur drei Patienten mit jeweils 5 Bildern zur Verfügung. Das bedeutet, dass es wenig Daten zum Trainieren und Prädizieren des Klassifikators gab und somit das Klassifikationsergebnis wie z.B. bei Patient A und B nicht sehr gut ausgefallen ist. Um die Qualität der Segmentierungen signifikant zu erhöhen würde ich in Zukunft mehr Patientendaten in die bestehende Softwareapplikation integrieren.

Des Weiteren ist mir aufgefallen, dass bei Durchführung des Real-Life-Tests die Einstreuung der Atembewegungen der Patienten in den CT-Bilddatensätzen deutlich zu sehen war. Dies machte sich dadurch bemerkbar, dass beim Übereinanderlegen der jeweiligen Bildmodalitäten eines Patienten das vom Mediziner eingezeichnete Tumorvolumen (der Goldstandard) nicht mit allen Bildaufnahmen eines Patienten übereinstimmte. So war beispielsweise in der venösen Bildaufnahme der Tumor um ein paar Voxel, gegenüber der nativen und arteriellen Bildaufnahme verschoben. Zukünftig könnte man die Erzeugung von Offsets in CT-Bilddatensätzen, die durch Atembewegungen verursacht worden sind, dadurch beheben, dass entsprechende Gegentransformationen berechnet werden. Beispielsweise nimmt man vor und nach der Ausatmung eine Bildaufnahme auf, registriert beide momentanen Aufnahmen und berechnet dadurch den Offset, der während einer Ein- und Ausatmungsphase zustande kommt. Anschließend zieht man den berechneten Offset von dem durch die Atembewegung verursachten Bilddatensatz ab und hätte so das Problem gelöst.

Eine weitere Schwierigkeit war, dass Venengänge fälschlicherweise von der Software als Tumorgewebe klassifiziert wurden. Beim Patienten B hat die entwickelte SVM die Tumorsegmentierung nur zu 2% richtig prädiziert. Ein möglicher Grund dafür ist, dass bei der Definierung der Trainingsmasken Voxelintensitäten (Grauwerte) als Tumorgewebe deklariert wurden, obwohl diese gar nicht zum Tumorbereich gehörten. Umständlich war allgemein die Definierung der Trainingsmasken im Real-Life-Test. Diese mussten sehr präzise definiert werden, denn sobald nur eine einzige falsche Voxelintensität, in die Segmentierung der Trainingsmaske aufgenommen wurde prädizierte der Klassifikator alles als Tumorbereich. Positiv anmerken kann man, dass die SVM auch für sehr große Mengen an Bilddaten sehr schnell klassifizieren kann, schneller als dies der Random-Forest-Klassifikator bei gleicher Datenmenge bewältigen könnte.

6.2 Ausblick

Wie bereits in der Diskussion erwähnt kann die entwickelte SVM zukünftig für poly phänotypische Tumore und nicht ausschließlich für Lebertumore eingesetzt werden. Auch der automatisierte Test könnte erweitert werden, sodass dieser die berechnete Accuracy fallspezifisch auswerten kann. Dies könnte beispielsweise dadurch erreicht werden, indem der automatisierte Test nach Berechnung der Accuracy die jeweiligen Dice-Scores berechnet und diese ins Verhältnis zueinander setzt. Zusätzlich ist es möglich eine entsprechende adaptierte GUI zu entwickeln, mit Hilfe derer man unter intuitiver Bedienung die SVM-Parameter setzen, die Kreuzvalidierung aktivieren bzw. deaktivieren und die Trainings- und Prädiktionszeiten pro Klassifkationsdurchlauf anzeigen lassen kann.

A. Anhang

A.1 Glossar

Merkmal	(engl. feature) ist im Kontext der Medizinischen Bildverarbeitung ein charakteristisches Merkmal in einem Bilddatensatz.
Modalität	bezeichnet in der Medizin verschiedene Medizingeräte, mit denen Bildserien aufgenommen werden können.
Pixel	(engl. picture element) entspricht einem Bildpunkt in einem zweidimensionalen Bild.
Voxel	(engl. volume element) ist ein dreidimensionales Pixel, welches einen Bildpunkt im dreidimensionalen Bilddatensatz beschreibt.
Matrix	beschreibt eine Anordnung von mathematischen Objekten. Medizinische Bilder können als zweidimensionale Matrizen interpretiert werden. Eine Matrix ist auch eine Anordnung von Vektoren.
Vektor	ist ein mathematisches Objekt, das durch seinen Betrag, die Richtung und den Anfangspunkt charakterisiert ist.
Vektorraum	ist eine algebraische Struktur, in der eine Menge von Vektoren enthalten ist.
Input space	(dt. Eingabevektorraum) bezeichnet eine Menge von Vektoren, die sich in einem niedrigdimensionalen Raum (z.B. im $\mathbb{R}^2$) befinden.
Feature space	(dt. Merkmalsvektorraum) beschreibt eine Menge von Vektoren, die sich in einem höherdimensionalen Raum (z.B. im $\mathbb{R}^5$) befinden.
Bias	bezeichnet in der Mathematik eine Art von Verzerrung. Wird synonym auch als Fehler bezeichnet.
Sample	wird in der Informatik als Stichprobe von Daten bezeichnet.
Hyperebene	(engl. hyperplane) ist mathematisch gesehen eine Ebene im $\mathbb{R}^n$, die Samples linear voneinander trennt.

Klasse	(engl. label) ist eine Architektur, bei der mehrere Objekte auf Grund von gleichen Merkmalen zusammengefasst werden.
Muster	auch Textur genannt, bedeutet die Quantifizierung einer Oberfläche, sodass numerische Werte die Oberfläche beschreiben.
Segmentierung	bezeichnet inhaltlich zusammenhängende Regionen, in welchen benachbarte Voxel oder Pixel zusammengefasst sind. Entspricht einer Hervorhebung von interessanten Strukturen.
Texel	ist ein Texturelement oder auch Texturpixel, das in einem Muster vorkommt.
Klassifikation	ist die Zuordnung eines Musters zu einer definierten Klasse.

A.2 XML-Datei

Auszug der verwendeten Collection.xml für die Datenintegration von Bildern

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!-- Beispiel XML für Datenintegration-->
<col name="" >
  <subcol name="9J4" description="Dummy Layer" id="9J4" >
    <data name="9J4-1" description="--" id="9J4-1" >
      <item name="nativ" filepath="" link="9J4-1/9J4-nativ-c.nii.gz" />
      <item name="nativg1.5" filepath="" link="9J4-1/9J4nativ-gaussian-15.nrrd" />
      <item name="nativdog1.5" filepath="" link="9J4-1/9J4nativ-dog-15.nrrd" />
      <item name="nativlog1.5" filepath="" link="9J4-1/9J4nativ-log-15.nrrd" />
      <item name="ven" filepath="" link="9J4-1/9J4-ven-c.nii.gz" />
      <item name="veng1.5" filepath="" link="9J4-1/9J4ven-gaussian-15.nrrd" />
      <item name="vendog1.5" filepath="" link="9J4-1/9J4ven-dog-15.nrrd" />
      <item name="venlog1.5" filepath="" link="9J4-1/9J4ven-log-15.nrrd" />
      <item name="art" filepath="" link="9J4-1/9J4-art-c.nii.gz" />
      <item name="artg1.5" filepath="" link="9J4-1/9J4art-gaussian-15.nrrd" />
      <item name="artdog1.5" filepath="" link="9J4-1/9J4art-dog-15.nrrd" />
      <item name="artlog1.5" filepath="" link="9J4-1/9J4art-log-15.nrrd" />
      <item name="GTV" filepath="" link="9J4-1/FinalSeg.nrrd" />
      <item name="liver" filepath="" link="9J4-1/9J4-liver-c.nii.gz" />
      </data>
  </subcol>
  <subcol name="DQA" description="Dummy Layer" id="DQA" >
    <data name="DQA-1" description="--" id="DQA-1" >
      <item name="nativ" filepath="" link="DQA-1/DQA-nativ-c.nii.gz" />
      <item name="nativg1.5" filepath="" link="DQA-1/DQAnativ-gaussian-15.nrrd" />
      <item name="nativdog1.5" filepath="" link="DQA-1/DQAnativ-dog-15.nrrd" />
      <item name="nativlog1.5" filepath="" link="DQA-1/DQAnativ-log-15.nrrd" />
      <item name="ven" filepath="" link="DQA-1/DQA-ven-c.nii.gz" />
      <item name="veng1.5" filepath="" link="DQA-1/DQAven-gaussian-15.nrrd" />
      <item name="vendog1.5" filepath="" link="DQA-1/DQAven-dog-15.nrrd" />
      <item name="venlog1.5" filepath="" link="DQA-1/DQAven-log-15.nrrd" />
      <item name="art" filepath="" link="DQA-1/DQA-art-c.nii.gz" />
      <item name="artg1.5" filepath="" link="DQA-1/DQAart-gaussian-15.nrrd" />
      <item name="artdog1.5" filepath="" link="DQA-1/DQAart-dog-15.nrrd" />
      <item name="artlog1.5" filepath="" link="DQA-1/DQAart-log-15.nrrd" />
      <item name="GTV" filepath="" link="DQA-1/FinalSeg2.nrrd" />
      <item name="liver" filepath="" link="DQA-1/DQA-liver-c.nii.gz" />
    </data>
  </subcol>
  <subcol name="73L" description="Dummy Layer" id="73L" >
    <data name="73L-1" description="--" id="73L-1" >
      <item name="nativ" filepath="" link="73L-1/73L-nativ-c.nii.gz" />
      <item name="nativg1.5" filepath="" link="73L-1/73Lnativ-gaussian-15.nrrd" />
      <item name="nativdog1.5" filepath="" link="73L-1/73Lnativ-dog-15.nrrd" />
      <item name="nativlog1.5" filepath="" link="73L-1/73Lnativ-log-15.nrrd" />
      <item name="ven" filepath="" link="73L-1/73L-ven-c.nii.gz" />
      <item name="veng1.5" filepath="" link="73L-1/73Lven-gaussian-15.nrrd" />
      <item name="vendog1.5" filepath="" link="73L-1/73Lven-dog-15.nrrd" />
      <item name="venlog1.5" filepath="" link="73L-1/73Lven-log-15.nrrd" />
      <item name="art" filepath="" link="73L-1/73L-art-c.nii.gz" />
      <item name="artg1.5" filepath="" link="73L-1/73Lart-gaussian-15.nrrd" />
      <item name="artdog1.5" filepath="" link="73L-1/73Lart-dog-15.nrrd" />
      <item name="artlog1.5" filepath="" link="73L-1/73Lart-log-15.nrrd" />
      <item name="GTV" filepath="" link="73L-1/FinalSeg2.nrrd" />
      <item name="liver" filepath="" link="73L-1/73L-liver-c.nii" />
    </data>
  </subcol>
</col>
```

B. LITERATURVERZEICHNIS

[ChLin 11] C.-C. Chang and C.-J. Lin,
"LIBSVM: a library for support vector machines."
ACM Transactions on Intelligent Systems and Technology,
2:27:1--27:27, 2011.
https://www.csie.ntu.edu.tw/~cjlin/libsvm (Stand 2011)

[LB 11] Liu, Bing,
"Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data."
Berlin, Heidelberg, Verlag Springer Berlin Heidelberg, 2011.
ISBN-13 978-3-642-19460-3

[HH 09] Heinz Handels,
„Medizinische Bildverarbeitung, Bildanalyse, Mustererkennung und Visualisierung für die computergestützte ärztliche Diagnostik und Therapie", Verlag Vieweg + Teubner , 2009.

[HK 07] Renate Huch und Klaus D. Jürgens,
„Mensch Körper Krankheit"
5. Auflage, Verlag Urban und Fischer, 2007

[CoVa 95] Corinna Cortes, Vladimir Vapnik,
„Support Vector Networks, Machine Learning",
20, 273-297 (1995) © 1995 Kluwer Academic Publishers, Boston.

[NFHS 11] Alfred Nischwitz, Max Fischer,
Peter Haberäcker, Gudrun Socher,
„Computergrafik und Bildverarbeitung Band 2", 3 Auflage Verlag Vieweg + Teubner 2011

[B.Jähne 12] Bernd Jähne,
„Digitale Bildverarbeitung und Bildgewinnung"
7. neu bearbeitete Auflage. 2012. – Springer Verlag

[CS RS 14] Catalin Stoean, Ruxandra Stoean,
"Support Vector Machines and Evolutionary Algorithms for Classification"
ISBN 978-3-319-06940-1, 2014.- Springer Verlag

[HS CW 15] Hildegard Speckmann, Cornelia Wittkowski,
„Handbuch Anatomie: Bau und Funktion des menschlichen Körpers",
Ausgabe 2015 Verlag h.f.ullmann

[LJ 07] Lucia Joel, Dissertation,
„Das Hepatozelluläre Karzinom im chirurgischen Krankengut-Resultate operativer und palliativer Therapieverfahren und bei Spontanverlauf“,
Eberhard-Karls-Universität zu Tübingen, 2007

[GBEB 13] Gesundheitsberichtserstattung des Bundes: Krebs in Deutschland (2009/2010) Kapitel 3.6 Leber S:42, Robert Koch Institut 9.Ausgabe 2013
http://www.rki.de/DE/Content/Gesundheitsmonitoring/Gesundheitsberichterstattung/GBEDownloadsB/KID2013.pdf?__blob=publicationFile
(Stand: 2013)

[IW VI 04] Ivo Wolf, Marcus Vetter, Hans-Peter Meinzer,
“The Medical Imaging Interaction Toolkit (MITK) -
a toolkit facilitating the creation of interactive software by extending VTK and ITK”
Deutsches Krebsforschungszentrum DKFZ Heidelberg 2004.

[LGSG 01] Linda G. Shapiro and George C. Stockman,
Computer Vision, Upper Saddle River:
Prentice–Hall, 2001

[TKAK 15] Trambitskiy K.V., Anding K., Polte G.A., Garten D., Musalimov V.M. (2015),
"Out-of-focus region segmentation of 2D surface images with the use of texture features."
http://ntv.ifmo.ru/en/article/13839/segmentaciya_nesfokusirovannyh_oblasteyna_2D-izobrazheniyah_poverhnostey_s_ispolzovaniemteksturnyh_priznakov.htm (Stand Nov-Dez 2015)

[Apo15] Apothekenumschau,
„Leberkrebs (Leberzellkrebs, Leberkarzinom, Hepatozelluläres Karzinom)“
http://www.apotheken-umschau.de/Leberkrebs
(Stand: 29 Dez 2015)

[AB JW 09] Asa Ben-Hur, Jason Weston,
„A User´s Guide to Support Vector Machines“
Data Mining Techniques for the Life Sciences 30 Oktober 2009
Springer Verlag